Los Alamos National Laboratory

Simple Classifiers

Adam Cannon
Department of Computer Science
Columbia University
New York, NY 10027, USA

James Howse, Don Hush and Clint Scovel
Modeling, Algorithms and Informatics Group, CCS-3
Mail Stop B265
Los Alamos National Laboratory
Los Alamos, NM 87545

LANL Technical Report: LA–UR–03–0193

Report Date: January 16, 2003

Abstract

In this paper we introduce simple classifiers as an example of how to use the data dependent hypothesis class framework in (Cannon, Ettinger, Hush, & Scovel, 2002a) to explore the performance/computation trade–off in the classifier design problem. We demonstrate that simple classifiers have many remarkable properties. For example they possess computationally efficient learning algorithms with favorable bounds on estimation error, admit kernel mappings, are particularly well suited to boosting, and are fully parallelizable. In addition they are robust to the choice of learning problem which we demonstrate with the error minimization, Neyman–Pearson and min–max problems. Our experiments with synthetic and real data suggest that simple classifiers are competitive with powerful alternative methods.

Acknowledgments

We would like to thank Pat Kelly and Mike Fugate for running some of the test problems in Section 5.

1 Introduction

Consider the standard machine learning framework built around the Vapnik–Chervonenkis (VC) theory (Vapnik, 1998) and its treatment of the well known *error minimization* problem where we seek a classifier that minimizes the expected classification error. This framework provides performance guarantees for classifiers designed through empirical error minimization, but empirical error minimization is computationally hard for most nontrivial hypothesis classes. On the other hand restriction to trivial hypothesis classes alleviates the computational difficulties but does not provide good performance. These two extremes emphasize the importance of developing frameworks that facilitate the exploration of more moderate portions of the performance/computation space. In (Cannon et al., 2002a) a modification of the standard framework was introduced that enables this type of exploration in a nontrivial way. It involves an extension of the VC theory to the case where the hypothesis class depends on the data. In this new framework classifier design is decomposed into two components: the first is a restriction to a *data dependent* subclass of the hypothesis class and the second is empirical error minimization within that subclass. Exploration of the performance/computation trade–off is then performed in terms of the choice of *data dependent hypothesis class*. The study of how this choice affects performance is decomposed into two terms: *estimation error* which quantifies that portion of the error due to finite sample effects and *approximation error* which is the best error achievable by the data dependent class. To distinguish between hypothesis classes that are data dependent and those that are not we refer to the latter as "traditional classes". (Cannon et al., 2002a) introduced and analyzed several elementary examples of data dependent classes that were shown to fall between the two extremes. These examples motivate the introduction of *simple classes*.

We call a data dependent hypothesis class *simple* if the specific class realized by any given data set is the polynomial union of linearly ordered hypothesis classes, where a linearly ordered class is one whose indicator sets are linearly ordered by subset inclusion. We show how this definition facilitates the development of bounds on estimation error and computation. It also appears to facilitate the discovery of classes which are expressive enough to have good approximation error in practice. For example, for each pair of points from the training set consider the class of linear classifiers defined by all hyperplanes orthogonal to the difference between the pair. We define the *linear point–to–point* (LPP) class to be the union of these classes over all such pairs. This simple class has some remarkable properties. For example, in contrast to the intractability of empirical error minimization over traditional linear classifiers, we derive a low order polynomial time algorithm for empirical error minimization over LPP. This algorithm is simple, numerically robust, fully parallelizable and has no free parameters. In addition we obtain bounds on estimation error for LPP that are independent of dimension and similar in form to those obtained for a traditional class with VC dimension equal to 3. In the process of proving this result we prove similar bounds on the difference between training and generalization error which is beneficial when it is important to have an accurate estimate of the actual performance of the classifier in the absence of a large test set. Because the estimation error bounds are independent of dimension we are motivated to map to a higher dimensional space to improve performance, and since LPP classifiers are linear, computations can still be performed in the original space by employing kernel mappings. This situation is similar to support vector machines but here we optimize empirical error over a restricted class rather

than margin over the unrestricted class. In summary, through an elementary construction, we are able to obtain the simple class LPP which when coupled with empirical error minimization successfully addresses nearly all the key issues in the classifier design problem.

The outstanding issue is approximation error, which we do not analyze theoretically but rather through simulations. Our results are consistent with the traditional framework in that they depend on how well the class is matched to the process generating the data. To demonstrate the ease with which the simple class framework can address this issue we use elementary constructions to create four additional simple classes. They all share the property that they admit computationally efficient algorithms for empirical error minimization. In addition they admit computationally efficient algorithms for *weighted* empirical error minimization. This is important because approximation error can be reduced through boosting which calls for the production of a base classifier at each round that minimizes a weighted empirical error. For most nontrivial hypothesis classes minimizing weighted empirical error is computationally intractable but for our five simple classes we show that it is not. Consequently these simple classes can be used to facilitate bonafide boosting algorithms. Our empirical results demonstrate that boosting simple classes works well. For example we show that an unboosted simple class which performs poorly can be boosted to a performance that is comparable to powerful methods such as support vector machines and random forests.

Although error minimization is the most common learning problem treated in the literature there are others that are very important in practice. For example, in the *Neyman–Pearson* problem we minimize class 0 error while constraining class 1 error below some fixed value. In the *min–max* problem the design criterion contains unknown free parameters and the problem consists of building a classifier which is robust to their future value. This is most commonly accomplished through the min–max procedure made popular in robust statistics by Huber (Huber, 1981). In (Cannon, Howse, Hush, & Scovel, 2002b) we analyze learning strategies and provide bounds on estimation error for both Neyman–Pearson and min–max over traditional classes. These learning strategies are computationally intractable for most nontrivial classes but we demonstrate that for simple classes they are not. Indeed we develop computationally efficient algorithms and favorable estimation error bounds for both Neyman–Pearson and min–max for all five simple classes. The ease with which this is accomplished suggests that the simple class framework is robust to the choice of learning problem.

This paper is outlined as follows. In Section 2 we define data dependent hypothesis classes, data dependent shatter coefficients and recall the bounds on estimation error obtained in (Cannon et al., 2002a) for empirical error minimization over data dependent hypothesis classes. In Section 3 we define the Neyman–Pearson and min–max learning problems and extend the results in (Cannon et al., 2002b) to derive bounds on estimation error for data dependent hypothesis classes. In Section 4 we define five simple classes, develop bounds on their shatter coefficients, show how to incorporate kernels, and develop computationally efficient algorithms for the error minimization, Neyman–Pearson and min–max learning strategies. In Section 5 we report results for experiments with both synthetic and real data.

2 Error Minimization over Data Dependent Hypothesis Spaces

Consider a set X, a finite set Y and a probability space $Z = X \times Y$. Let $z = (x, y)$ denote the corresponding random variable with probability measure $\mathcal{P}$, conditional probability measures $\mathcal{P}_y$ and y-marginal probability measure P. Let $\mathcal{F}$ denote a class of functions (classifiers) $f : X \to Y$ and let

$$e(f) = \mathcal{P}(f(x) \neq y)$$

denote the generalization error of the classifier f. Let $e^* = \inf_{f \in \mathcal{F}} e(f)$ denote the best error achievable in the class $\mathcal{F}$. We further suppose that we collect n independent identically distributed (*i.i.d.*) samples $(z(1), z(2), .., z(n))$ from $\mathcal{P}$ and use them to construct an empirical error function

$$\hat{e}(f) = \frac{1}{n} \sum_{i=1}^{n} I\big(f(x(i)) \neq y(i)\big).$$

The work of Vapnik and Chervonenkis (Vapnik & Chervonenkis, 1974) justifies the time honored learning strategy that chooses $\hat{f}$ to minimize $\hat{e}$ [1] by establishing the following probabilistic guarantee when $Y = \{0, 1\}$:

$$\mathcal{P}^n(e(\hat{f}) - e^* > \epsilon) \leq 8n^{V(\mathcal{F})} e^{-n\epsilon^2/128}, \tag{1}$$

where $V(\mathcal{F})$ is the Vapnik–Chervonenkis dimension of the function class $\mathcal{F}$. For this paper it is important to note that this result follows from their (Vapnik & Chervonenkis, 1974) bound

$$\mathcal{P}^n(\sup_{f \in \mathcal{F}} |e(f) - \hat{e}(f)| > \epsilon) \leq 8n^{V(\mathcal{F})} e^{-n\epsilon^2/32} \tag{2}$$

on error deviance combined with their lemma

$$e(\hat{f}) - e^* \leq 2 \sup_{f \in \mathcal{F}} |e(f) - \hat{e}(f)|. \tag{3}$$

We refer to (3) as a *fundamental lemma* because it bounds estimation error in terms of error deviance thereby transforming results on the convergence of empirical processes to results in learning theory.

These results are applicable only when the the hypothesis class is chosen *before* data is observed. In (Cannon et al., 2002a) it was shown that one could allow the hypothesis class to depend on the data and still obtain a bound on estimation error similar to (1). Following (Cannon et al., 2002a) we outline this result now. Let z_n denote the n-sample consisting of individual samples $z_n(i), i = 1, .., n$. Given an n-sample z_n, we consider functions from a hypothesis space $\mathcal{F}_{z_n}$ which can depend on the n-sample and so is defined by a class $\mathcal{F}_n$

[1] Throughout this paper we ignore questions of whether minima or maxima are actually attained. This detail is easy to include by introducing approximation parameters and approximate minima/maxima but obscures the presentation.

of functions on (Z^n, Z). We define a data dependent class $\mathcal{F} = \{\mathcal{F}_n\}$ to be a collection of such classes $\mathcal{F}_n$. Next we define shatter coefficients for data-dependent function classes. Even though we consider Y other than the binary $Y = \{0, 1\}$, for the purposes of this paper we only need to define the shatter coefficients for $\{0, 1\}$–valued function classes.

Definition 2.1. For $n \leq m + k_1$ define $\mathcal{N}(z_{k_1}, z_m, \mathcal{F}_n)$ to be the number of distinct dichotomies of the m points z_m generated by the function classes $\mathcal{F}_{z_n}$ where $z_n = z_{k_1} \cup z_{k_0}$ and z_{k_0} varies over all subsets $z_{k_0} \subset z_m$ with $k_0 = n - k_1$. That is the number of different sets in

$$\{\{z_m(1), ..., z_m(m)\} \cap I_f : f \in \mathcal{F}_{z_n}, z_n = z_{k_1} \cup z_{k_0}, z_{k_0} \subset z_m\}.$$

We define the shatter coefficients

$$S_{k_0,m}(\mathcal{F}_n) = \sup_{z_{k_1}, z_m} \mathcal{N}(z_{k_1}, z_m, \mathcal{F}_n).$$

Elementary considerations show that the function S is monotonically increasing in its two subscript indices and therefore for any $\acute{n} \leq n$

$$S_{\acute{n},2\acute{n}}(\mathcal{F}_n) \leq S_{n,2n}(\mathcal{F}_n). \tag{4}$$

Note that these shatter coefficients are more general than those in (Cannon et al., 2002a). In the notation of that paper $S_{n/m}(\mathcal{F}) = S_{n,m}(\mathcal{F}_n)$.

In (Cannon et al., 2002a) the following theorem was mentioned.

Theorem 2.1. *Let $Y = \{0, 1\}$ and let $\mathcal{F}$ be a data-dependent classifier space. Given an i.i.d. n-sample z_n, let*

$$e^*_{z_n} = \inf_{f \in \mathcal{F}_{z_n}} e(f)$$

denote the optimal generalization error in the data dependent class $\mathcal{F}_{z_n}$ and choose $\hat{f}$ to solve the learning strategy

$$\min_{f \in \mathcal{F}_{z_n}} \hat{e}(f) \tag{5}$$

of minimizing the empirical error over the class $\mathcal{F}_{z_n}$. Then for any $m > n$ and ϵ,

$$\mathcal{P}^n(e(\hat{f}) - e^*_{z_n} > \epsilon) \leq 2S_{n,m}(\mathcal{F}_n)e^{\epsilon}e^{-\frac{1}{4}(\frac{1}{n}+\frac{1}{m-n})^{-1}\epsilon^2}.$$

Proof. First utilize the main result from (Cannon et al., 2002a)

$$\mathcal{P}^n(\sup_{f \in \mathcal{F}_{z_n}} |e(f) - \hat{e}(f)| > \epsilon) \leq 2S_{n,m}(\mathcal{F}_n)e^{2\epsilon}e^{-(\frac{1}{n}+\frac{1}{m-n})^{-1}\epsilon^2} \tag{6}$$

which bounds the error deviance. Then apply a data dependent version (Cannon et al., 2002a) ,

$$e(\hat{f}) - e^*_{z_n} \leq 2 \sup_{f \in \mathcal{F}_{z_n}} |e(f) - \hat{e}(f)|, \tag{7}$$

of the fundamental lemma (3), bounding the estimation error in terms of error deviance, to complete the proof.

♦

3 Alternative Learning Problems

We return for the moment to the framework where $\mathcal{F}$ is a traditional class and describe three well–known classifier design problems. The first is the *error minimization* problem

$$\min_{f \in \mathcal{F}} e(f) \tag{8}$$

which was the subject of the previous section. The second is the *Neyman–Pearson* problem. Let $Y = \{0, 1\}$ and consider the class conditional errors e_i defined by

$$e_i(f) = \mathcal{P}_i(f(x) \neq i), \ i \in Y.$$

The Neyman–Pearson problem is motivated by real world scenarios where it is important that one of these errors be no greater than some fixed value. For example in fraud detection the classification system often has no utility unless the false alarm rate (i.e. the rate at which fraudulent activity is predicted when it is not present) can be kept below a fixed level. In the Neyman–Pearson problem we impose this type of constraint on one of the errors and optimize the other. The version of the Neyman–Pearson problem we treat here is (e.g. see (Van-Trees, 1968))

$$\begin{array}{ll} \min_{f \in \mathcal{F}_1} & e_0(f) \\ \text{where} & \mathcal{F}_1 = \{f : f \in \mathcal{F}, e_1(f) \leq \alpha\}. \end{array} \tag{9}$$

The third problem we consider is the *min–max* problem. This problem is motivated by real world scenarios where the error e is defined in terms of an unknown parameter q and we wish to design a classifier that is robust to its value. In the min–max problem the classifier is determined by solving

$$\min_{f \in \mathcal{F}} \max_{q \in Q} e(f, q). \tag{10}$$

A common example is where $Y = \{0, 1\}$ and the unknown parameter is the class marginal $q = P(y = 0)$. By unknown we mean that $P(y = 0)$ is not known ahead of time nor do we get information about it in the sample data. This situation can occur either because the value of the class marginal may change in the future or because the sample data is gathered in such a way that the number of samples from each class are not determined by the random process generating the data.

For each of these three problems we can formulate learning strategies which specify how to select a classifier $\hat{f}$ given an n-sample z_n. For example, for the error minimization problem it is common to choose $\hat{f}$ to solve the learning strategy

$$\min_{f \in \mathcal{F}} \hat{e}(f) \tag{11}$$

of minimizing the the empirical error determined by the n-sample z_n. Regardless of the strategy, a standard benchmark for its performance is the estimation error. For the error minimization problem the estimation error is

$$e(\hat{f}) - e^* \tag{12}$$

where $e^* = \inf_{f \in \mathcal{F}} e(f)$. For the min–max problem we define

$$e^* = \min_{f \in \mathcal{F}} \max_{q \in Q} e(f, q) \tag{13}$$

and then define the estimation error to be

$$\max_{q \in Q} e(\hat{f}, q) - e^*. \tag{14}$$

For the Neyman-Pearson problem it is not so clear how to proceed. We propose that a reasonable definition for the estimation error is the 2-tuple

$$\big(e_0(\hat{f}) - \min_{f \in \mathcal{F}_1} e_0(f), e_1(\hat{f}) - \alpha\big). \tag{15}$$

In (Cannon et al., 2002b) it was shown how to formulate learning strategies for both the Neyman-Pearson problem and the min–max problem for which bounds on estimation error could be obtained. This was accomplished by combining the development of fundamental lemmas that bound estimation error in terms of error deviance for these strategies (similar to (3) for the error minimization strategy) with bounds on error deviance (such as the VC theorem (2)). Here we extend these results to data dependent hypothesis spaces. To this end, observe that in (Cannon et al., 2002a) it was noted that the data dependent fundamental lemma (7) for empirical error minimization follows easily from the proof of the data *in*-dependent fundamental lemma (3). In much the same way the data dependent versions of the fundamental lemmas for both the Neyman-Pearson and the min–max problem follow from the proofs of the data *in*-dependent versions found in (Cannon et al., 2002b). If we combine these fundamental lemmas with the bounds (6) on error deviance we can obtain bounds on estimation error for both the Neyman-Pearson problem and min–max problem utilizing these learning strategies over data dependent hypothesis spaces. These bounds depend on the sample plan, i.e. the method used to gather training data. In the theorems below we provide estimation error bounds for both problems assuming a retrospective sample plan where $n_0 > 0$ samples are drawn *i.i.d.* from $\mathcal{P}_0$, $n_1 > 0$ samples are drawn *i.i.d.* from $\mathcal{P}_1$, and $n_0 + n_1 = n$. Bounds for the Neyman-Pearson problem can also be obtained for the sample plan where n samples are drawn *i.i.d.* from $\mathcal{P}$, but we do not present them here. The estimation error bound for the Neyman-Pearson problem is the following.

Theorem 3.1. *Consider the retrospective sample plan where we choose $n_0 > 0$ samples with $y = 0$ i.i.d. from $\mathcal{P}_0$ and $n_1 > 0$ samples with $y = 1$ i.i.d. from $\mathcal{P}_1$ where $n = n_0 + n_1$. Denote the distribution for this plan $\mathcal{P}_S = \mathcal{P}_0^{n_0} \mathcal{P}_1^{n_1}$. Given an n-sample z_n collected according to this sample plan, let*

$$\begin{array}{ll} e^*_{0,z_n} = \min_{f \in \mathcal{F}_{1,z_n}} & e_0(f) \\ \textit{where} & \mathcal{F}_{1,z_n} = \{f : f \in \mathcal{F}_{z_n}, e_1(f) \leq \alpha\} \end{array} \tag{16}$$

denote the Neyman-Pearson value for the data dependent class $\mathcal{F}$ at z_n. Also define the empirical class errors to be

$$\hat{e}_i(f) = \frac{1}{n_i} \sum_{j : y_n(j) = i} I(f(x_n(j)) \neq i). \tag{17}$$

Choose $\epsilon_1 > 0$ and then choose $\hat{f}$ to solve the learning strategy

$$\begin{array}{ll} \min_{f \in \hat{\mathcal{F}}_{1,z_n}} & \hat{e}_0(f) \\ \textit{where} & \hat{\mathcal{F}}_{1,z_n} = \{f : f \in \mathcal{F}_{z_n}, \hat{e}_1(f) \leq \alpha + \epsilon_1/2\}. \end{array} \tag{18}$$

Then for any $\epsilon_0 > 0$,

$$\mathcal{P}_S\Big(\big(e_0(\hat{f}) - e^*_{0,z_n} > \epsilon_0\big) \textit{ or } \big(e_1(\hat{f}) > \alpha + \epsilon_1\big)\Big) \leq$$

$$6 S_{n,2n}(\mathcal{F}_n)(e^{-n_0\epsilon_0^2/8} + e^{-n_1\epsilon_1^2/8}).$$

Proof. The fundamental lemma for the Neyman-Pearson problem (Lemma 1 in (Cannon et al., 2002b)) can be easily extended to its data dependent version providing a data dependent version of Corollary 1 from (Cannon et al., 2002b). That is

$$\mathcal{P}_S\Big(\big(e_0(\hat{f}) - e^*_{0,z_n} > \epsilon_0\big) \text{ or } \big(e_1(\hat{f}) > \alpha + \epsilon_1\big)\Big) \leq$$

$$\mathcal{P}_S\Big(\sup_{f \in \mathcal{F}_{z_n}} |e_0(f) - \hat{e}_0(f)| > \epsilon_0/2\Big) + \mathcal{P}_S\Big(\sup_{f \in \mathcal{F}_{z_n}} |e_1(f) - \hat{e}_1(f)| > \epsilon_1/2\Big).$$

Consider the first term on the righthand side. With z_{n_1} fixed, $\mathcal{F}_{z_n} = \mathcal{F}_{z_{n_0}, z_{n_1}}$ is a z_{n_0} dependent class so that we can generalize Theorem 3 from (Cannon et al., 2002a) (6) to the class errors and obtain

$$\mathcal{P}_0^{n_0}\Big(\sup_{f \in \mathcal{F}_{z_n}} |e_0(f) - \hat{e}_0(f)| > \epsilon_0/2\Big) \leq 2 S_{n_0,2n_0}(\mathcal{F}_n) e^{\epsilon_0} e^{-n_0\epsilon_0^2/8}.$$

Since $\mathcal{P}_S = \mathcal{P}_0^{n_0}\mathcal{P}_1^{n_1}$ and the supremum with respect to the z_{n_1} variable is utilized in the definition of the shatter coefficients we obtain

$$\mathcal{P}_S\Big(\sup_{f \in \mathcal{F}_{z_n}} |e_0(f) - \hat{e}_0(f)| > \epsilon_0/2\Big) \leq 2 S_{n_0,2n_0}(\mathcal{F}_n) e^{\epsilon_0} e^{-n_0\epsilon_0^2/8}$$

with a similar statement for the term $\mathcal{P}_S\Big(\sup_{f \in \mathcal{F}_{z_n}} |e_1(f) - \hat{e}_1(f)| > \epsilon_1/2\Big)$. Consequently

$$\mathcal{P}_S\Big(\big(e_0(\hat{f}) - e^*_{0,z_n} > \epsilon_0\big) \text{ or } \big(e_1(\hat{f}) > \alpha + \epsilon_1\big)\Big) \leq$$

$$2 S_{n_0,2n_0}(\mathcal{F}_n) e^{\epsilon_0} e^{-n_0\epsilon_0^2/8} + 2 S_{n_1,2n_1}(\mathcal{F}_n) e^{\epsilon_1} e^{-n_1\epsilon_1^2/8}.$$

We now use $e^\epsilon \leq 3$ and the inequality (4) to finish the proof. ♦

Our next theorem gives a bound on estimation error for the min–max problem for multiclass classification when the unknown q is the class marginals.

Theorem 3.2. *Let Y be a finite set with $|Y| = M$ and let Q be the space of all probability measures on Y. Consider a data dependent function class $\mathcal{F}$ of Y-valued functions and let $\mathcal{F}^i, i \in Y$ denote the data dependent classes of binary functions i or not i derived from $\mathcal{F}$ in the natural way. Consider the case where the free parameters $q \in Q$ in*

$$e(f,q) = \sum_i e_i(f) q_i$$

are the Y marginals. Consider a retrospective sample plan S of size n with $n_i > 0$ samples drawn i.i.d. from $\mathcal{P}_i$ and $\sum_i n_i = n$. Denote the probability measure for this sample plan $\mathcal{P}_S$. Given an n-sample z_n, let

$$e^*_{z_n} = \min_{f \in \mathcal{F}_{z_n}} \max_{q \in Q} e(f,q) \tag{19}$$

denote the min–max value for the data dependent class $\mathcal{F}$ at z_n and choose $\hat{f}$ to solve the learning strategy

$$\min_{f \in \mathcal{F}_{z_n}} \max_{i \in Y} \hat{e}_i(f) \tag{20}$$

with the empirical class errors $\hat{e}_i$ defined in (17). Then for any $q \in Q$,

$$\mathcal{P}_S\Big(e(\hat{f},q) - e^*_{z_n} > \epsilon\Big) \le 6M \max_{i \in Y} S_{n,2n}(\mathcal{F}^i_n) e^{-n_i \epsilon^2/8}. \tag{21}$$

Proof. The fundamental lemma for the the min-max problem(Lemma 3 in (Cannon et al., 2002b)) can be easily extended to its data dependent version providing a data dependent version of Corollary 2 of (Cannon et al., 2002b). That is

$$\mathcal{P}_S\Big(e(\hat{f},q) - e^*_{z_n} > \epsilon\Big) \le \mathcal{P}_S\Big(\sup_{f \in \mathcal{F}_{z_n}, q \in Q} |e(f,q) - \hat{e}(f,q)| > \epsilon/2\Big).$$

Since

$$\sup_{q \in Q} |e(f,q) - \hat{e}(f,q)| \le \sup_{i \in Y} |e_i(f) - \hat{e}_i(f)|$$

we can apply the union bound followed by the data dependent bound (6) on the error deviance applied to the class errors and utilize the above definitions of shatter coefficients to obtain

$$\mathcal{P}_S\Big(e(\hat{f},q) - e^*_{z_n} > \epsilon\Big) \le 2Me^{\epsilon} \max_{i \in Y} S_{n_i,2n_i}(\mathcal{F}^i_n) e^{-n_i \epsilon^2/8}.$$

Application of $e^{\epsilon} \le 3$ and the inequality (4) finishes the proof. ♦

4 Examples of Simple Classifiers

In this section we analyze five data dependent hypothesis classes that are representative of data dependent hypothesis classes that we call simple classes. Three of the five are restrictions of traditional linear classifiers and two are restrictions of traditional spherical classifiers. Two of the five classes are shown to possess shatter coefficient bounds that are independent of dimension (establishing a tight bound for the other three is an open problem). For computational considerations we let $X = \mathbb{R}^d$ with the usual inner product $\langle \cdot, \cdot \rangle$ and metric $\| \cdot \|$. In all five cases we present polynomial–time algorithms for all three empirical problems in (5), (18) and (20).

Recall that $z_n = (x_n, y_n)$ and let $\Xi_{z_n} = \{\xi\}$ be a finite set of *representation points* $\xi \in \mathbb{R}^d$ determined by a *representation set rule* applied to an n–sample z_n. Each of our simple classes takes the form

$$\mathcal{F}_{z_n} = \{f : f(x) = \mathcal{H}\big(s(g(x, \xi) - b)\big),\ s \in \{-1, 1\},\ b \in \mathbb{R},\ \xi \in \Xi_{z_n}\} \tag{22}$$

where $\mathcal{H}(\alpha) = \left\{ \begin{smallmatrix} 1 & \text{if } \alpha > 0 \\ 0 & \text{if } \alpha \leq 0 \end{smallmatrix} \right.$ is the heaviside function and $g(\cdot, \cdot)$ is the *structure function* determining the type of classifier (*i.e.* linear or spherical). A total of five simple classes are obtained by employing two structure functions, one with three representation set rules and the other with two. The two options for g are the linear structure function $g(x, \xi) = \langle\, x, \xi\, \rangle$, and the spherical structure function $g(x, \xi) = \|x - \xi\|$. The five simple classes are determined by (22) and the five data dependency rules that follow.

1. The *linear point–to–point* (LPP) data dependency rule is defined by the linear structure function and the representation set rule

$$\Xi_{z_n} = \{\xi : \xi = x_n(i) - x_n(j),\ 1 \leq i < j \leq n\}. \tag{23a}$$

2. The *linear point–to–centroid* (LPC) data dependency rule is defined by the linear structure function and the representation set rule

$$\Xi_{z_n} = \{\xi : \xi = x_n(i) - \mu_n,\ 1 \leq i \leq n\}, \tag{23b}$$

where

$$\mu_n = \frac{1}{n} \sum_{i=1}^{n} x_n(i).$$

3. The *linear centroid–to–centroid* (LCC) data dependency rule is defined by the linear structure function and the representation set rule

$$\Xi_{z_n} = \{\xi : \xi = \mu_{n_0} - \mu_{n_1}\}, \tag{23c}$$

where

$$\mu_{n_i} = \begin{cases} \dfrac{1}{n_i} \displaystyle\sum_{j : y_n(j) = i} x_n(j) & \text{if } n_i > 0, \\ 0 & \text{if } n_i = 0, \end{cases} \qquad i = 0, 1.$$

4. The *sphere about a point* (SP) data dependency rule is defined by the spherical structure function and the representation set rule

$$\Xi_{z_n} = \{\xi : \xi = x_n(i),\ 1 \le i \le n\}. \tag{23d}$$

5. The *sphere about the centroid* (SC) data dependency rule is defined by the spherical structure function and the representation set rule

$$\Xi_{z_n} = \{\xi : \xi = \mu_n\}, \tag{23e}$$

where

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_n(i).$$

We use the same five acronyms LPP, LPC, LCC, SP and SC to refer to the simple classes defined by these rules.

When a data dependent class $\mathcal{F}$ is obtained by restricting a traditional class $\mathfrak{F}$ then

$$S_{n,2n}(\mathcal{F}_n) \le (2n)^{VC(\mathfrak{F})} + 1.$$

Consequently(see (Devroye, Györfi, & Lugosi, 1996)), for the linear classes LPP, LPC, and LCC we have

$$S_{n,2n}(\mathcal{F}_n) \le (2n)^{d+1} + 1$$

and for the spherical classes SP, and SC we have

$$S_{n,2n}(\mathcal{F}_n) \le (2n)^{d+2} + 1.$$

These bounds may be very loose. Indeed, in (Cannon et al., 2002a) it was shown that the shatter coefficients for the LPP and SP classes satisfy $S_{n,2n}(\mathcal{F}_n) \le 8n^3 - 2n$ and $S_{n,2n}(\mathcal{F}_n) \le 4n^2$ respectively, both of which are independent of dimension. Although the LPC, LCC, and SC classes appear to be less complex, establishing a tight bound on their shatter coefficients is currently an open problem.

It is interesting to note that for LPP and SP the shatter coefficient satisfies (see (Cannon et al., 2002a), p. 348)

$$S_{n,2n}(\mathcal{F}_n) \le (2n)^{VC_{2n}(\mathcal{F})} + 1$$

where $VC_{2n}(\mathcal{F})$ is the data dependent VC dimension of order $2n$ and is defined as

$$VC_{2n}(\mathcal{F}) = \max_{\{n\,:\,\exists\ z_n \subseteq z_{2n}\,:\,\mathcal{F}_{z_{2n}}\ \text{shatters}\ z_n\}} n.$$

In words, $VC_{2n}(\mathcal{F})$ is the size of the largest subset of some $2n$ points which is shattered by the data dependent class on those $2n$ points. When the data dependent class $\mathcal{F}$ is obtained by restricting a traditional class $\mathfrak{F}$ then $VC_{2n}(\mathcal{F})$ will be less than or equal to the VC dimension $VC(\mathfrak{F})$.

We now describe a simple polynomial time algorithm that optimizes all three empirical problems in (5), (18) and (20) over all five data dependent classes in (23). Because we will be interested in boosting these classifiers we describe a more general algorithm that optimizes generalizations of these three problems in which each of the n data points $z_n(i)$ is given some positive weight $w_n(i)$. The empirical weighted class error, which is a generalization of (17), is given by

$$\hat{e}_i(f) = \frac{1}{n_i} \sum_{j\,:\,y_n(j)=i} w_n(j)\; I\big(f(x_n(j)) \neq i\big) \tag{24}$$

where $w_n(i) \geq 0, \;\; \forall \;\; i = 1, \ldots, n$. This definition of $\hat{e}_0$ and $\hat{e}_1$ is used in (5), (18) and (20) to define the empirical weighted error, empirical weighted Neyman-Pearson and empirical weighted min-max problems. Note that defining $w_n(i) = 1, \;\forall \;\; i = 1, \ldots, n$ gives the standard unweighted problems.

The algorithm we describe optimizes over the set of triplets $(\xi, s, b) \in \Xi_{z_n} \times \{-1, 1\} \times \mathbb{R}$ which parameterizes all five data dependent classes. Therefore we adopt the notation $\hat{e}_i(\xi, s, b)$ for the empirical weighted class error of the function $f \in \mathcal{F}_{z_n}$ parameterized by (ξ, s, b). All three empirical problems in (5), (18) and (20) can be solved as minimization problems of the form [2]

$$\min_{\xi,s,b} L\big(\hat{e}_0(\xi, s, b), \hat{e}_1(\xi, s, b)\big)$$

where the loss function L is defined by

$$L(a_0, a_1) = \left(\frac{n_0}{n}\right) a_0 + \left(\frac{n_1}{n}\right) a_1 \qquad \text{for error minimization,} \tag{25a}$$

$$L(a_0, a_1) = \begin{cases} a_0, & a_1 \leq \alpha + \epsilon_1/2 \\ \infty, & \text{otherwise} \end{cases} \qquad \text{for Neyman-Pearson and} \tag{25b}$$

$$L(a_0, a_1) = \max(a_0, a_1) \qquad \text{for min-max.} \tag{25c}$$

We use the property $\min_{\xi,s,b} = \min_{\xi} \min_{s,b}$ to decompose the algorithm into a loop that minimizes over the finite set Ξ_{z_n} and a computation inside the loop that minimizes over the infinite set $\{-1, 1\} \times \mathbb{R}$. This is shown in Algorithm 1 which computes the representation set Ξ_{z_n} and then loops over all representation points, calling the `MinThresholdSign` routine on line 7 to solve the optimization problem $\min_{s,b} L\big(\hat{e}_0(\xi, s, b), \hat{e}_1(\xi, s, b)\big)$ for each representation point. The `MinThresholdSign` routine returns a solution (s_ξ, b_ξ) and the value l_ξ of the loss function at that solution, and lines 8–10 track a globally optimal solution.

To develop the algorithm for the `MinThresholdSign` routine let ξ be fixed and define the discriminant values

$$v_n(j) = g(x_n(j), \xi), \;\; j = 1, ..., n \tag{26}$$

so that

$$f(x_n(j)) = \mathcal{H}\big(s(v_n(j) - b)\big).$$

[2] Since each of the five data dependent classes contains at least one function that correctly classifies all class 1 samples there is always a feasible solution to the Neyman–Pearson problem in (18).

Algorithm 1 `Learning Algorithm for Simple Classifiers`

`INPUTS:` $z_n,\ w_n, L$
`OUTPUTS:` (ξ^*, b^*, s^*)

Initialize $l^* = \infty$ and $(\xi^*, s^*, b^*) = (0, 1, 0)$.
$\Xi_{z_n} \leftarrow$ `ComputeRepresentationSet` (z_n)
for all $\xi \in \Xi_{z_n}$ **do**
  $\big(l_\xi, s_\xi, b_\xi\big) \leftarrow$ `MinThresholdSign`$\big(L, \xi, z_n, w_n\big)$
  if $(l_\xi < l^*)$ **then**
    $(l^*, \xi^*, s^*, b^*) \leftarrow (l_\xi, \xi, s_\xi, b_\xi)$
  end if
end for

return $\{(\xi^*, b^*, s^*)\}$

When ξ is fixed and $s = 1$ the family of indicator sets $X(b) = \{x_n(j) : f(x_n(j)) = \mathcal{H}\big(v_n(j) - b\big) = 1\}$ is linearly ordered by subset inclusion which means that the weighted empirical errors $\hat{e}_0$ and $\hat{e}_1$ are monotonic in b. This linear ordering also means that there are at most $n + 1$ indicator sets (including the empty set) and therefore the error pair $(\hat{e}_0, \hat{e}_1)$, and consequently each loss function in (25), takes on at most $n + 1$ distinct values. Indeed, if $K \leq n + 1$ is the number of distinct discriminant values and we let $v'(1) < v'(2) < ... < v'(K)$ be the ordered list of these values, then the number of sets is equal to $K + 1$ and they are witnessed by any set $\{b_0, b_1, ..., b_K\}$ whose members satisfy

$$b_k \in \big[v'(k), v'(k+1)\big),\ k = 0, 1, ..., K \tag{27}$$

where we define $v'(0) = -\infty$ and $v'(K + 1) = \infty$. The same is true when $s = -1$ except that the members of $\{b_0, b_1, ..., b_K\}$ must satisfy

$$b_k \in \big(v'(k), v'(k+1)\big],\ k = 0, 1, ..., K \tag{28}$$

which differ from (27) only at the boundaries. Now let $\delta > 0$ and define $v'(0) = v'(1) - \delta$ and $v'(K + 1) = v'(K) + \delta$ so that the specific set $\{b_0, b_1, ..., b_K\}$ with members

$$b_k = \frac{v'(k) + v'(k+1)}{2}$$

satisfies both (27) and (28). With this set the problem $\min_{s,b} L\big(\hat{e}_0(\xi, s, b), \hat{e}_1(\xi, s, b)\big)$ can be solved by determining a member of $\{-1, 1\} \times \{b_0, b_1, ..., b_K\}$ with the minimum loss value. We develop an efficient algorithm for computing the loss values for all members of this set by exploiting the property that $\hat{e}_0$ and $\hat{e}_1$ are monotonic in b . Specifically, once the value of $\hat{e}_i$ has been computed for (s, b_k) this property allows us to compute the value for (s, b_{k+1}) with a single addition. Our algorithm also exploits the fact that once $\hat{e}_i$ has been determined for $s = 1$ its value for $s = -1$ is simply $c_i - \hat{e}_i$, where $c_i = \frac{1}{n_i} \sum_{j:y_n(j)=i} w_n(j)$. Thus, once the values of $\hat{e}_0$ and $\hat{e}_1$ have been initialized for $(s, b) = (1, b_0)$ it is a simple matter to step through all the members of $\{-1, 1\} \times \{b_0, b_1, ..., b_K\}$ in order, updating $\hat{e}_0$ and $\hat{e}_1$ and computing the loss values using (25) as we go.

The complete `MinThresholdSign` routine is illustrated in Algorithm 2. The values c_0 and c_1 which are used to compute weighted empirical errors for $s = -1$ from weighted empirical errors for $s = 1$ are computed on line 4. Lines 6-8 compute the discriminant values v_n and line 10 produces a sorted index list σ for the discriminant values, i.e. σ is such that $v_n(\sigma(j)) \leq v_n(\sigma(j+1)),\ j = 1, ..., n-1$. Lines 13-33 compute loss values for all members of $\{-1, 1\} \times \{b_0, b_1, ..., b_K\}$ and save a pair (s_ξ, b_ξ) with the smallest loss value l_ξ. The starting point $(s, b) = (1,\ v_n(\sigma(1)) - \delta)$, its weighted empirical error values $\hat{e}_0$ and $\hat{e}_1$, and corresponding loss value are determined on lines 13-14. Lines 15-17 treat the case where the initial b value is paired with $s = -1$. Then the loop in lines 19-33 steps through the remaining values of (s, b), updating the weighted empirical error values on lines 21-22, and computing and tracking the smallest loss on lines 24-31. Because v_n may contain repeated discriminant values we employ the "if" statement on line 24 to ensure that only values of b from the set $\{b_0, b_1, ..., b_K\}$ (i.e. values fall between distinct discriminant values) are considered.

The overall run time of Algorithm 1 is $O\big(\mathcal{T}_{CRS} + |\Xi_{z_n}|\mathcal{T}_{MTS}\big)$ where $\mathcal{T}_{CRS}$ is the run time of the `ComputeRepresentationSet` routine, $\mathcal{T}_{MTS}$ is the run time of the `MinThresholdSign` routine, and

$$|\Xi_{z_n}| \leq \begin{cases} \frac{n^2-n}{2}, & \text{LPP} \\ n, & \text{LPC} \\ 1, & \text{LCC} \\ n, & \text{SP} \\ 1, & \text{SC} \end{cases}$$

from (23). The `ComputeRepresentationSet` routine requires $\mathcal{T}_{CRS} = n^2 d$ time for the LPP class, $\mathcal{T}_{CRS} = nd$ time for LPC, LCC and SC and $\mathcal{T}_{CRS} = 0$ time for SP. The `MinThresholdSign` requires $\Theta(n)$ time [3] to compute c_0 and c_1 on line 4, a time $\mathcal{T}_{DISC}$ to compute the discriminant values on lines 6–8, $O(n \log n)$ time to perform the sort on line 10 and $\Theta(n)$ time to step through the values of (s, b) in lines 13–33. Since both the linear and the spherical structure functions can be written in terms of inner products of vectors in $\mathbb{R}^d$, the computational requirements for lines 6–8 are $\mathcal{T}_{DISC} = \Theta(nd)$. Thus, the run time of the `MinThresholdSign` routine is $O(nd + n \log n)$. Now, since the inequality $\mathcal{T}_{CRS} < |\Xi_{z_n}|\mathcal{T}_{MTS}$ holds for all five classes, the overall run time of Algorithm 1 is

$$O(|\Xi_{z_n}|(nd + n \log n)).$$

Thus, for the five data dependent classes in (23) the run time is at most $O(n^3 \max(d, \log n))$ and at best $O(n \max(d, \log n))$.

It is worth mentioning that the $n^2 d$ storage requirements for Ξ_{z_n} for the LPP class can be avoided by simply computing the representation point ξ as needed inside the loop in Algorithm 1. This is easily accomplished without increasing the run time. It is also worth mentioning that it is more efficient to move the computation of c_0 and c_1 from the `MinThresholdSign` routine to the initialization step on line 4 in Algorithm 1. Finally, our experience has shown that the set of triplets $\Xi_{z_n} \times \{-1, 1\} \times \{b_0, b_1, ..., b_K\}$ visited by Algorithm 1 often contains more than one member that minimizes the loss. Let θ be the subset of optimal triplets for a

[3] The notation $\mathcal{T} = \Theta(\alpha)$ means that $\mathcal{T}$ is bounded above and below by a linear function of α.

Algorithm 2 `MinThresholdSign`: This routine solves the optimization problem $\min_{s,b} L(\hat{e}_0(\xi,s,b),\hat{e}_1(\xi,s,b))$ and returns a solution (s_ξ, b_ξ) and the value l_ξ of the loss function at that solution.

`INPUTS`: $L, \xi, z_n = (x_n, y_n), w_n$
`OUTPUTS`: (l_ξ, b_ξ, s_ξ)

$c_0 \leftarrow \frac{1}{n_0}\sum_{j:y_n(j)=0} w_n(j)$, $c_1 \leftarrow \frac{1}{n_1}\sum_{j:y_n(j)=1} w_n(j)$

for $j = 1$ to n **do**
  $v_n(j) = g(x_n(j), \xi)$
end for

$\sigma \leftarrow$ `AscendingSort` (v_n)
Set $\sigma(n+1) \leftarrow n+1$ and $v_n(n+1) = v_n(\sigma(n)) + \delta$ for some small $\delta > 0$.

$\hat{e}_0 \leftarrow c_0$, $\hat{e}_1 \leftarrow 0$
$(l_\xi, s_\xi, b_\xi) \leftarrow \big(L(\hat{e}_0, \hat{e}_1),\ 1,\ v_n(\sigma(1)) - \delta\big)$
if $(L(c_0 - \hat{e}_0, c_1 - \hat{e}_1) < l_\xi)$ **then**
  $(l_\xi, s_\xi, b_\xi) \leftarrow \big(L(c_0 - \hat{e}_0, c_1 - \hat{e}_1),\ -1,\ v_n(\sigma(1)) - \delta\big)$
end if

for $j = 1$ to n **do**

  $\hat{e}_0 \leftarrow \hat{e}_0 - I(y_n(\sigma(j)) = 0)\left(\frac{w_n(\sigma(j))}{n_0}\right)$
  $\hat{e}_1 \leftarrow \hat{e}_1 + I(y_n(\sigma(j)) = 1)\left(\frac{w_n(\sigma(j))}{n_1}\right)$

  if $\big(v_n(\sigma(j)) \neq v_n(\sigma(j+1))\big)$ **then**
    if $(L(\hat{e}_0, \hat{e}_1) < l_\xi)$ **then**
      $(l_\xi, s_\xi, b_\xi) \leftarrow \left(L(\hat{e}_0, \hat{e}_1),\ 1,\ \left(\frac{v_n(\sigma(j)) + v_n(\sigma(j+1))}{2}\right)\right)$
    end if
    if $(L(c_0 - \hat{e}_0, c_1 - \hat{e}_1) < l_\xi)$ **then**
      $(l_\xi, s_\xi, b_\xi) \leftarrow \left(L(c_0 - \hat{e}_0, c_1 - \hat{e}_1),\ -1,\ \left(\frac{v_n(\sigma(j)) + v_n(\sigma(j+1))}{2}\right)\right)$
    end if
  end if

end for

return $\{(l_\xi, b_\xi, s_\xi)\}$

given problem instance. Algorithm 1 chooses from this set by simply keeping the first one it sees. However we may wish to choose differently. For example it may be more robust to choose randomly from θ. Alternatively we may wish to choose a solution that optimizes a criterion based on margin. These choices can be implemented with negligible computational cost.

Algorithm 1 has many attractive properties. First, it exactly minimizes the empirical error in a finite number of steps. Second, it uses only multiplications, additions and comparisons and is therefore likely to be more robust to finite precision computations than than other learning algorithms that employ operations such as division, linear solvers, and iterative solvers. Third, it has no free parameters for the user to specify and analyze. This algorithm is also trivial to parallelize by partitioning Ξ_{z_n} into q sets, each of size roughly $\frac{|\Xi_{z_n}|}{q}$, where q is the number of processors, and then running Algorithm 1 on each partition in parallel. Results of the parallel runs are coalesced by a $\Theta(q)$ algorithm to determine a global optimum.

We turn now to the design and implementation of simple *kernel* classifiers. Let $\phi : X \to \bar{X}$ be a map with a kernel $K(x_1, x_2)$ such that $\langle\, \phi(x_1), \phi(x_2) \,\rangle = K(x_1, x_2)$, where the dimension $\bar{d}$ of $\bar{X}$ may be much larger than d. Since both the linear and spherical structure functions $g(x, \xi) = \langle\, x, \xi \,\rangle$ and $g(x, \xi) = \|x - \xi\|$ can be expressed in terms of inner products it is straightforward to implement simple classifiers in the mapped space $\bar{X}$ without having to compute in $\bar{X}$ by using the kernel to compute inner products. Algorithm 1 is easily modified to accommodate these classifiers. Representation points are represented implicitly, i.e. by the data samples from z_n that determine them, so the explicit computation of the representation set Ξ_{z_n} on line 5 is omitted. The main loop iterates over all representations and computation of the discriminant values on line 7 of Algorithm 2 must be modified as shown below

LPP: with $\xi = \phi(x_n(i_1)) - \phi(x_n(i_2))$,
$$v_n(j) = K(x_n(j), x_n(i_1)) - K(x_n(j), x_n(i_2))$$

LPC: with $\xi = \phi(x_n(i)) - \bar{\mu}_n$,
$$v_n(j) = K(x_n(j), x_n(i)) \;-\; \underbrace{\frac{1}{n}\sum_{l=1}^{n} K(x_n(j), x_n(l))}_{\gamma(j)}$$

LCC: with $\xi = \bar{\mu}_{n_0} - \bar{\mu}_{n_1}$,
$$v_n(j) = \underbrace{\frac{1}{n_0}\sum_{l:y_n(l)=0} K(x_n(j), x_n(l))}_{\gamma_0(j)} \;-\; \underbrace{\frac{1}{n_1}\sum_{l:y_n(l)=1} K(x_n(j), x_n(l))}_{\gamma_1(j)}$$

SP: with $\xi = \phi(x_n(i))$,
$$v_n(j) = \sqrt{K(x_n(j), x_n(j)) - 2\,K(x_n(j), x_n(i)) + K(x_n(i), x_n(i))}$$

SC: with $\xi = \bar{\mu}_n$,
$$v_n(j) = \sqrt{K(\boldsymbol{x}_n(j), \boldsymbol{x}_n(j)) \;-\; 2\,\underbrace{\frac{1}{n}\sum_{l=1}^{n} K(\boldsymbol{x}_n(j), \boldsymbol{x}_n(l))}_{\gamma(j)} \;+\; \underbrace{\frac{1}{n^2}\sum_{l_1=1}^{n}\sum_{l_2=1}^{n} K(\boldsymbol{x}_n(l_1), \boldsymbol{x}_n(l_2))}_{\gamma_2}}$$

where $\bar{\mu}_n$, $\bar{\mu}_{n_0}$ and $\bar{\mu}_{n_1}$ are centroids of the mapped data. Let κ be the number of computations required to perform a kernel evaluation. Then the time required to compute all n discriminant

values using the formulas above is $\Theta(n\kappa)$ for LPP and SP, $\Theta(n^2\,\kappa)$ for LPC and LCC, and $\Theta(n^3\,\kappa)$ for SC. This is in contrast to the nd time reported for the *non*–kernelized algorithm. However, by computing the quantities $\gamma(j), \gamma_0(j), \gamma_1(j), j = 1, ..., n$ and γ_2 (as needed) at the initialization step on line 4 of Algorithm 1 we can reduce the time required to compute the discriminant values in Algorithm 2 to $n\kappa$. With these modifications the kernelized version of Algorithm 1 has run time $O(|\Xi_{z_n}|(n\kappa + n\log n))$, except for the SC class whose run time is now $O(n^2\kappa)$ because it is dominated by the computation of γ_2 during the initialization step. Note that the run time of the kernelized algorithm can be reduced to $O(|\Xi_{z_n}|\, n\, \log n)$ for the LPP, LPC and SP classes by initially computing the array M whose elements are $M_{ij} = K(x_n(i), x_n(j))\ \ \forall\ \ 1 \leq i < j \leq n$ at the cost of $\Theta(n^2)$ in additional storage.

Algorithm 1 also has some very attractive properties when coupled with boosting. Recall that boosting is a procedure that, given a training set, produces an overall classifier that is a weighted majority vote of classifiers from a base hypothesis class. Since boosting is a procedure that is employed after we see the data it is unchanged by the utilization of a data dependent base hypothesis class as long as this class is the same for each round of boosting. Most boosting strategies call for the determination of a base classifier at each round that minimizes a weighted empirical error. However, most classifier design algorithms used in practice *do not* minimize empirical error because it is computationally intractable for the function classes they consider (*e.g.*, traditional linear classifiers), and so satisfaction of this objective is rarely guaranteed. On the other hand, Algorithm 1 is guaranteed to minimize weighted empirical error when boosting the simple classifiers in (23). In addition, the run time for all boosting rounds after the first can be improved by saving information computed during the first round. In successive rounds of boosting Algorithm 1 is invoked with the same data z_n but different weights w_n. Thus by saving the discriminant values v_n and the sorted index list σ computed on lines 6–11 of Algorithm 2 for each ξ during the first boosting round we can avoid these computations on future rounds. This reduces the run time to $\Theta(|\Xi_{z_n}|n)$ for each boosting round after the first. This run time speed–up requires an additional storage of $\Theta(|\Xi_{z_n}|n)$.

5 Experimental Results

The development of practical methods for classifier design involves a trade–off between computation, estimation error and approximation error. In the previous section we developed an algorithm that implements a computationally efficient learning strategy for three learning problems (error minimization, Neyman–Pearson and min-max) over five simple classes: LPP, LPC, LCC, SP and SC. The shatter coefficient bounds for these learning strategies give rise to favorable estimation error bounds in all fives cases, particularly for the LPP and SP classes where the bound is independent of dimension. Indeed, in this section we demonstrate that these learning strategies are highly resistant to overfitting, even when the dimension is high and the sample size is small. The advantages of computational efficiency and favorable estimation error must be balanced against a possible sacrifice in approximation error. In principle, in the data dependent hypothesis framework, approximation error is controlled through the choice of data dependency. Ideally this choice is based on first principles knowledge of the problem at hand. However, in this paper we have no specific problem in mind so we have chosen five data dependencies that illustrate the flexibility and diversity of this approach and we test their

performance on both synthetic and real data sets.

An appealing characteristic of the learning strategies for the five basic classes is that they contain no free parameters. Employing a kernel map however, adds one free parameter (i.e. the choice of kernel) and boosting adds another (i.e. the number of boosting rounds). Employing a kernel map is attractive in that it results in a very minor change in the computational requirements, and does not affect the estimation error bounds for the LPP and SP classes. On the other hand boosting increases the computational requirements and may also increase the estimation error. Nevertheless the overall computational requirements for boosting remain practical and empirical studies have suggested that it can be very effective at reducing approximation error without overfitting (Schapire, Freund, Bartlett, & Lee, 1998). In addition these simple classes possess special properties (described in the previous section) that make them particularly well suited to boosting. Our current application of boosting is limited to the error minimization problem since we know of no boosting strategy for the Neyman–Pearson or min–max problems.

5.1 Synthetic Data

Our first set of experiments are performed on synthetic data generated from Gaussian distributions where we can compute the optimal classifiers and the values for the error minimization, Neyman-Pearson, and min-max problems in (8), (9) and (10). Each of these three problems possesses a *generalization value* $\mu(f)$ when evaluated at a particular classifier f. For the error minimization problem $\mu(f) = e(f)$, for the Neyman-Pearson problem we choose the two–tuple $\mu(f) = (e_0(f), e_1(f))$, and for the min-max problem $\mu(f) = \max_i e_i(f)$. We refer to the generalization value at the optimal measurable classifier as the Bayes' value μ^*. In our experiments we report estimates of the average generalization value $E_S[\mu(\hat{f})] = \int \mu(\hat{f}) d\mathcal{P}_S$ of the learning strategy, where the average is over all training sets of a given size n generated according to the sampling plan S. We employ the *i.i.d.* sample plan for the error minimization problem (where n samples are drawn independent and identically distributed from $\mathcal{P}$) and the retrospective sample plan for the Neyman–Pearson and min–max problems (where $n_0 > 0$ samples are drawn *i.i.d.* from $\mathcal{P}_0$, $n_1 > 0$ samples are drawn *i.i.d.* from $\mathcal{P}_1$, and $n_0 + n_1 = n$). In addition, for the min-max problem we report estimates of the average error $E_S[e(\hat{f}, q)] = qE_S[e_0(\hat{f})] + (1-q)E_S[e_1(\hat{f})]$ achieved by the learning strategy as the future class marginal $q = P(y = 0)$ ranges from 0 to 1.

We compare our simple classifier learning strategies with the Gaussian Maximum Likelihood (GML) learning strategy (e.g. see (Fukunaga, 1990)). We choose GML because it has similar computational requirements and is one of the few alternative learning strategies that accommodates all three learning problems in a natural way. In addition it is a time honored method whose generalization properties are well known, particularly for the problem instances we have chosen for our experiments. Indeed, we expect the GML learning strategy to perform well in these experiments because it exploits the knowledge that the conditional class distributions are Gaussian. With this knowledge the optimal classifier for all three learning problems can be expressed as a quadratic classifier of the form

$$\mathcal{H}\Big((x - m_1) \cdot \Sigma_1^{-1}(x - m_1) - (x - m_0) \cdot \Sigma_0^{-1}(x - m_0) + \ln\left(|\Sigma_1|/|\Sigma_0|\right) + \tau\Big) \tag{29}$$

where the m_i and Σ_i are the conditional class means and covariances and the threshold τ is chosen so that $\tau = 2\ln(P(y=1)/P(y=0))$ for error minimization, $e_1 = \alpha$ for Neyman–Pearson and $e_0 = e_1$ for min–max. The GML learning strategy produces a quadratic classifier by computing maximum likelihood estimates of m_i, Σ_i, substituting them into (29), and then using maximum likelihood estimates of $\mathcal{P}(y=i)$ to determine τ for error minimization, and empirical error estimates of e_i to determine τ for the Neyman–Pearson and min–max problems. To make this strategy complete we must specify its course of action when the estimated covariance matrices have reduced rank. In addition we would like to be robust to cases where the estimated covariance matrices are ill–conditioned. We address these concerns by inverting regularized covariance estimates of the form $\hat{\Sigma}_i + \gamma I$ where $\hat{\Sigma}_i$ is the maximum likelihood estimate and $\gamma > 0$ is the *regularization parameter* for our GML strategy. The computational requirements for this strategy are $O(nd^2 + d^3 + n\log n)$.

For the error minimization problem we also compare our simple classifier learning strategy with support vector machines. We choose support vector machines because, like the learning strategies for LPP and SP, they possess estimation error bounds that are independent of dimension (Cristianini & Shawe-Taylor, 2000) and can be solved by a computationally efficient learning algorithm (e.g. see (Hush & Scovel, 2003)). In particular we employ the soft margin support vector machine (SVM–SM) described in (Cortes & Vapnik, 1995) with the specific SVM^{light} learning algorithm described in (Joachims, 1999). This learning strategy has three parameters: The kernel function K, the positive real value C that weights the slack variables in the SVM–SM criterion, and a positive real value *tol* associated with the stopping condition for the algorithm [4]. The run times of SVM–SM algorithms tend to be more variable than the run times of simple classifier algorithms or GML algorithms. SVM–SM algorithms are often sensitive to C (i.e. larger values of C typically lead to longer run times) and the properties of the data (i.e. less separable data typically leads to longer run times). Although the run time guarantees for practical SVM-SM algorithms are as high as $O(C^2n^5\log n)$ (Hush & Scovel, 2003), empirical evidence suggests that the run time dependence on n tends to be no higher than cubic (Campbell & Cristianini, 1999; Joachims, 1999; Williams & Seeger, 2001).

In our synthetic data experiments we set $d = 50$ and generate data according to probability distributions on $\mathbb{R}^d \times \{0,1\}$ where the class conditional distributions $\mathcal{P}_0$ and $\mathcal{P}_1$ are Gaussian. We perform experiments with two different distributions whose parameters are chosen so that the Bayes' value for the error minimization problem is 0.15. Specifically, in the first distribution the class means are equal $\mu_0 = \mu_1 = 0$ and the class covariances are different $\Sigma_0 = 1.467I$ and $\Sigma_1 = I$, and in the second distribution the class covariances are equal $\Sigma_0 = \Sigma_1 = I$ and the class means are different $\mu_0 = (0.276)1$ and $\mu_1 = 0$. Following (Fukunaga, 1990) we refer to the first distribution as the I–γI distribution and the second as the I–I distribution. For both distributions the class marginal probabilities are $P(y=0) = 1/3, P(y=1) = 2/3$ for the error minimization problem (recall that class marginals are not part of the Neyman–Pearson and min–max problems). The corresponding Bayes' values for the three design problems are summarized in Table 1.

Fukunaga (1990) suggests that practical learning strategies should be general enough to perform well on both of these distributions because in many classification problems the charac-

[4] The iterative algorithm in (Joachims, 1999) is terminated when the first order necessary conditions for an optimum are satisfied within a tolerance *tol*.

	Error Minimization	Neyman-Pearson ($\alpha = 0.10$)	Min-Max
Bayes' Value for I–γI	0.150	(0.254, 0.100)	0.171
Bayes' Value for I–I	0.150	(0.251, 0.100)	0.164

Table 1: Bayes' values for the synthetic data experiments.

teristic that distinguishes the two classes is some combination of mean difference and covariance difference. These particular distributions were selected because many learning strategies which work well on the I–γI distribution do not work well on the I–I distribution and vice versa. He points out that although the covariance matrices in both these distributions are diagonal, they are still representative because any two non-diagonal covariance matrices can be simultaneously diagonalized by a linear transformation provided that at least one of them is positive definite.

When judging the performance of a learning strategy we are interested in the average generalization value of the learning strategy, where the average is over training sets of a given size n. We compute unbiased estimates of these average generalization values as follows. For the error minimization problem the training and test sets described below are generated using *i.i.d.* sampling and for the min-max and Neyman-Pearson problems they are generated retrospectively with an equal number of samples per class. For each problem (error minimization, Neyman-Pearson and min-max), and each applicable learning strategy (e.g. LPP, SVM-SM, GML, boosting, etc.), and each training sample size $n = 10, 30, 50, 100, 250, 500$ the following is repeated $k = 20$ times. First, n samples are randomly generated and used to train a classifier $\hat{f}$. Next, the errors $e(\hat{f})$, $e_0(\hat{f})$ and $e_1(\hat{f})$ of the resulting classifier are estimated using a test set of 50,000 random samples [5]. Finally, the estimated errors are averaged over the $k = 20$ runs to obtain estimates of the average errors $E_S[e(\hat{f})]$, $E_S[e_0(\hat{f})]$ and $E_S[e_1(\hat{f})]$. These are used to report estimates of the average generalization values $E_S[\mu(\hat{f})] = E_S[e(\hat{f})]$ for the error minimization problem and $E_S[\mu(\hat{f})] = \big(E_S[e_0(\hat{f})], E_S[e_1(\hat{f})]\big)$ for the Neyman–Pearson problem. The quantity $\max(e_0(\hat{f}), e_1(\hat{f}))$ is averaged over the $k = 20$ runs to obtain an estimate of average generalization value $E_S[\mu(\hat{f})] = E_S[\max(e_0(\hat{f}), e_1(\hat{f}))]$ for the min–max problem. In addition, the estimates of $E_S[e_0(\hat{f})]$ and $E_S[e_1(\hat{f})]$ are used to report the average error $E_S[e(\hat{f}, q)] = qE_S[e_0(\hat{f})] + (1-q)E_S[e_1(\hat{f})]$ for the min–max problem. Although the values we report are "estimates of averages", for expository purposes we drop the terms *estimate* and *average* and simply refer to them as the *errors* or *generalization values*.

We begin with the experimental results for the error minimization problem. The results for the I–γI distribution are illustrated in Figures 1(a)–1(d). In Figures 1(a)–1(c) the Bayes' value of $e^* = 0.15$ is shown as a dark dotted line. The generalization values for the three simple classes, LPP, SP and SC (without kernels or boosting), are plotted as a function of n in Figure 1(a). The generalization values for the other two linear classes, LPC and LCC, are similar to LPP so we show only the result for LPP. These results provide a rather dramatic example of how the choice of data dependency can affect approximation error. On one hand no linear classifier can perform well on this problem and so any class that is a restriction of linear

[5] The same test set of 50,000 *i.i.d.* random samples is used for all error minimization experiments and the same test set of 50,000 random samples generated retrospectively is used for all min-max and Neyman-Pearson experiments.

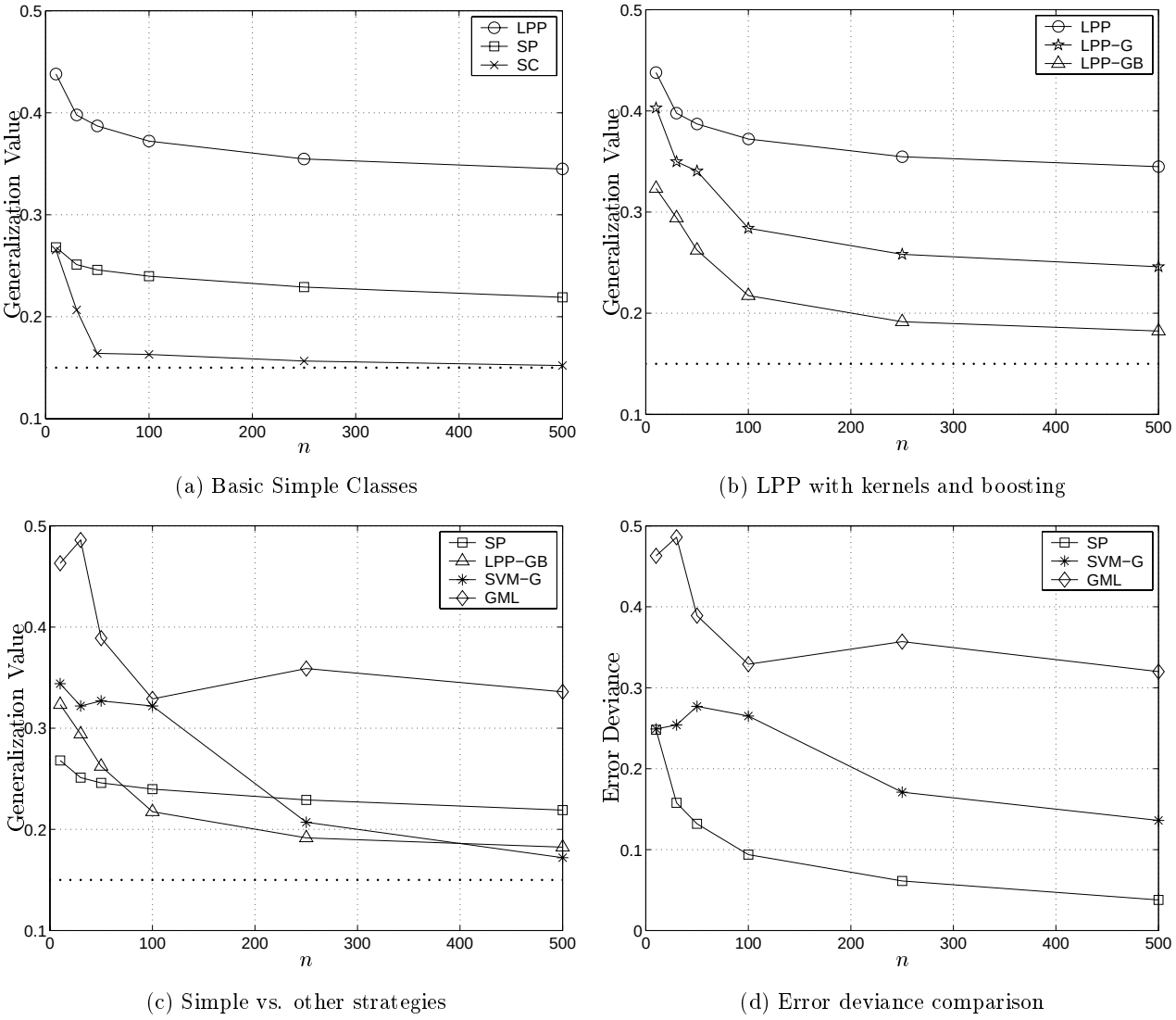
classifiers will necessarily perform poorly. On the other hand spherical classifiers are perfectly matched to this problem (since the optimal decision boundary is spherical) and since the SP and SC classes are restrictions of spherical classifiers it is not surprising that their performance is superior to LPP. Because the SP and SC classes are different restrictions however, we see a substantial difference in their generalization values. Since the generalization value for the SC class very nearly achieves the Bayes' value for such small training sample sizes we conclude that this choice of data dependency is particularly well suited to this problem. Indeed, except for the choice of threshold τ, the learning strategy employed for the SC class on this particular problem is identical to a GML strategy that exploits knowledge that the covariance matrices are of the form γI and the means are equal. Thus the SC classifier represents a close approximation to what we might do if such an abundance of first principles knowledge were available.

Figure 1(b) illustrates how the performance of a simple class that is not particularly well suited to this data distribution can be improved by adding a kernel and boosting. This figure plots the generalization value as a function of n for LPP, LPP with a Gaussian kernel (LPP-G) and LPP with Gaussian kernel and 500 rounds of boosting (LPP-GB). The Gaussian kernel is $K(x_1, x_2) = e^{-\|x_1 - x_2\|^2/d}$ and LPP-GB uses the AdaBoost method described in Freund and Shapire (1997). These results demonstrate that the approximation error can be dramatically reduced by incorporating kernels and boosting. Indeed the kernel alone improves the performance by approximately 10%. Boosting adds another 6% making the performance of LPP-GB superior to SP.

In Figure 1(c) we compare simple classifiers to some powerful alternative learning strategies. This figure plots the generalization value as a function of n for GML, SVM-SM, SP and LPP-GB [6]. The regularization parameter for GML was $\gamma = 10^{-6}$. For the support vector machine we used a Gaussian kernel (same as above) and the default values of C and tol in SVM^{light} (i.e. $C = n/\left(\sum_{i=1}^{n} K(x_n(i), x_n(i))\right)$ and $tol = 0.001$). The results here are somewhat surprising! First, the SVM-SM, SP and LPP-GB strategies all perform very well compared to GML even though they are not explicitly designed to exploit the knowledge that the probability distributions are Gaussian. Second, the SP and LPP-GB strategies attain much lower generalization values than SVM-SM for the smaller sample sizes. This demonstrates that by a suitable choice of data dependency (e.g. SP) or by kernels and boosting (e.g. LPP-GB) the simple classifier strategy is able to attain generalization values that are superior to some very powerful alternative methods. It is important to point out that these comparisons would be much different for larger sample sizes. Indeed, the performance of SVM-SM is already superior to the simple classes at $n = 500$ and for sufficiently large n the performance of GML will surpass that of the simple classifiers. Nevertheless it is striking that simple classifiers are so clearly dominant at these smaller sample sizes.

These results are obtained with a simple classifier strategy that is more *stable* than GML and SVM-SM. We use the term *stable* to refer to the sensitivity of the performance of the learning strategy to changes in the training data. Formal definitions of stability are described in (Bousquet & Elisseeff, 2002) where they are used to study the performance of support vector machines. In particular they show how to obtain bounds on the average classifier error deviance $E_S[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of stability so that learning strategies with good stability

[6] Although the results for the simple class SC are superior to all others we exclude it from this comparison because it exploits far more knowledge of the distribution than these other methods.



(a) Basic Simple Classes

(b) LPP with kernels and boosting

(c) Simple vs. other strategies

(d) Error deviance comparison

Figure 1: This figure summarizes the experimental results for the error minimization problem with the I–γI distribution. Figure 1(a) plots the generalization values for the simple classes LPP, SP and SC. Figure 1(b) plots the generalization values for LPP, LPP with Gaussian kernel (LPP-G) and LPP with Gaussian kernel and 500 rounds of boosting (LPP-GB). Figure 1(c) plots the generalization values for GML, SVM–SM with Gaussian kernel (SVM–G), SP and LPP-GB. In Figures 1(a)–1(c) the Bayes' value of $e^* = 0.15$ is shown as a dark dotted line. Figure 1(d) plots the error deviance $E_S[|e(\hat{f}) - \hat{e}(\hat{f})|]$ for GML, SVM–G and SP.

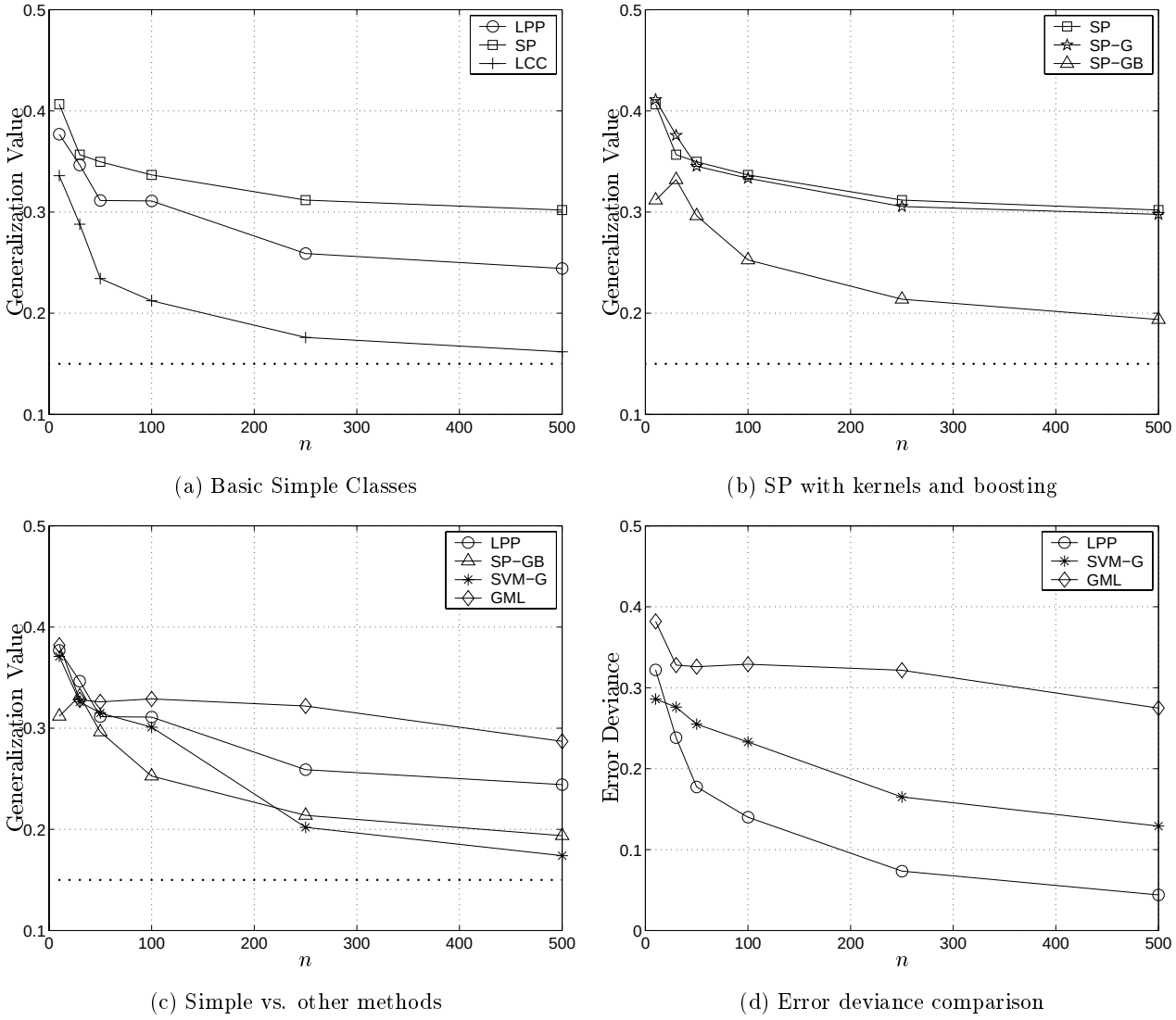
have small error deviance. In Figure 1(d) we plot (an estimate of) the average classifier error deviance $E_S[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of n for SP, GML and SVM-SM. The error deviance is substantially smaller for the SP learning strategy. Such results are predicted by the performance bounds established earlier in this paper where we produced favorable bounds on estimation error as a by-product of favorable bounds on error deviance. This characteristic of the simple classifier learning strategy means that we have higher confidence in our estimates of generalization value.

More precisely it means that the true generalization value is known to fall within a smaller range with higher confidence. This is particularly useful in applications where a large test set is not available and it is important to have an accurate estimate of performance.

Results for the same set of experiments with the I–I distribution are illustrated in Figures 2(a)–2(d). The Bayes' optimal decision boundary for this distribution is linear so we expect the restrictions of linear classifiers to perform best. Indeed the plot in Figure 2(a) demonstrates that the performances of the simple linear classes LPP and LCC are superior to the simple spherical class SP. In this plot we also see a substantial difference in the generalization values of the two different restrictions of linear classifiers LPP and LCC. Figure 2(b) illustrates how the performance of the simple class SP, which is not particularly well suited to this data distribution, is affected by the addition of a kernel and boosting. Incorporating a Gaussian kernel does not make much difference, but boosting improves the performance substantially and yields a performance that is superior to LPP. Figure 2(c) compares the simple classifier learning strategies LPP and SP-GB to the alternative strategies GML and SVM-SM. The parameters for the GML and SVM-SM learning strategies are the same as our first experiment. Again we see that the SVM-SM, LPP and SP-GB strategies all perform very well compared to GML. With this distribution however, the performance of SVM-SM surpasses that of simple classes at smaller training set sizes. Even so the performance of SP-GB is competitive with SVM-SM. These results again demonstrate that by a suitable choice of data dependency (e.g. LPP) or by kernels and boosting (e.g. SP-GB) the simple classifier strategy is competitive with powerful alternative methods. In Figure 2(d) we plot (an estimate of) the average classifier error deviance $E_S[|e(\hat{f}) - \hat{e}(\hat{f})|]$ as a function of n for LPP, GML and SVM-SM. Again we see that the error deviance is substantially smaller for the simple class LPP.

We now turn to the experimental results for the Neyman–Pearson and min–max problems. Much of what we have learned about simple classifiers from the experiments above carry over to our experiments with the Neyman–Pearson and min–max problems, so we present only the results comparing simple classifiers to alternative learning strategies. Also, since we know of no boosting or SVM strategy for these problems we compare only simple classifiers (with and without kernels) and GML. In addition, since the same two distributions are used in these experiments the simple classes that perform well here tend to be the same ones that performed well on the error minimization problem. Therefore in the experiments with the I–γI distribution we present results for the SP and LPP-G simple classes and in the experiments with the I–I distribution we present results for the LPP and SP-G simple classes [7]. We start with the Neyman–Pearson problem. For this experiment we set the class 1 error bound to $\alpha = 0.1$ and the accuracy parameter to $\epsilon_1 = 0.02$ (for a review of these parameters see the Neyman–Pearson learning strategy in (18)). Figure 3 presents the results for the I–γI distribution. Figures 3(a) and 3(b) plot the two components of the generalization value $(E_S[e_0(f)], E_S[e_1(f)])$ as a function of n for GML, SP and LPP-G. Except for the $n = 100$ case the class 1 errors for the three strategies are similar, but the class 0 errors are significantly smaller for the simple classifiers. To obtain a receiver operating curve (ROC) we set $n = 500$ and employed the GML, SP and LPP-G learning strategies with values of α from 0 to 1 in increments of 0.05, with $\epsilon_1 = 0.02$ in each case. Figure 3(c) plots the corresponding sequence of two-tuples (class 1

[7] The SC and LCC classes perform extremely well on the I–γI and I–I distributions respectively, but they are excluded in our comparisons because they exploit far more knowledge of the distributions than the other methods.



(a) Basic Simple Classes

(b) SP with kernels and boosting

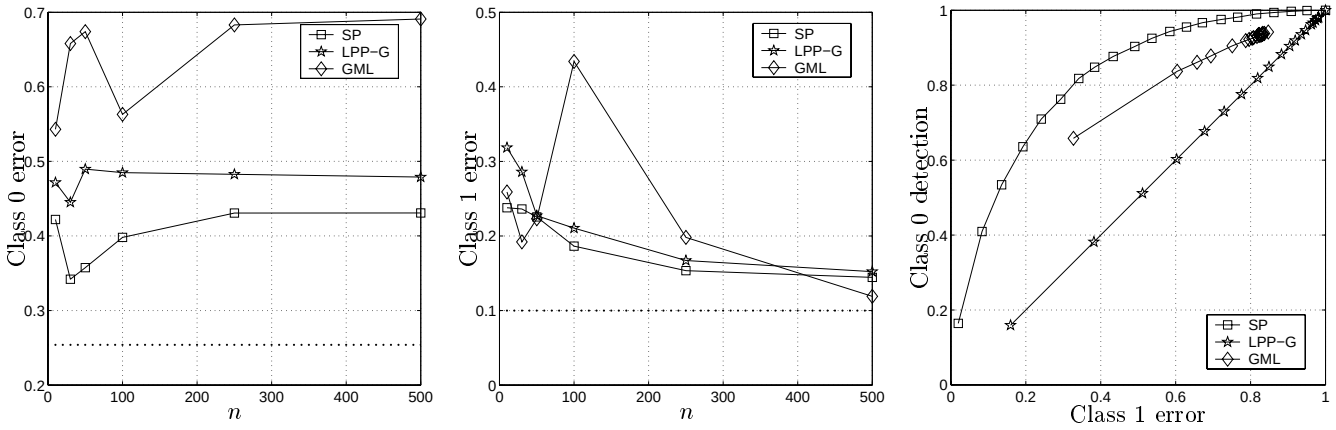
(c) Simple vs. other methods

(d) Error deviance comparison

Figure 2: This figure summarizes the experimental results for the error minimization problem with the *I–I* distribution. Figure 1(a) plots the generalization values for the simple classes SP, LPP and LCC. Figure 1(b) plots the generalization values for SP, SP with Gaussian kernel (SP-G) and SP with Gaussian kernel and 500 rounds of boosting (SP-GB). Figure 1(c) plots the generalization values for GML, SVM–SM with Gaussian kernel (SVM–G), LPP and SP-GB. In Figures 1(a)–1(c) the Bayes' value of $e^* = 0.15$ is shown as a dark dotted line. Figure 1(d) plots the error deviance $E_S[|e(\hat{f}) - \hat{e}(\hat{f})|]$ for GML, SVM–G and LPP.

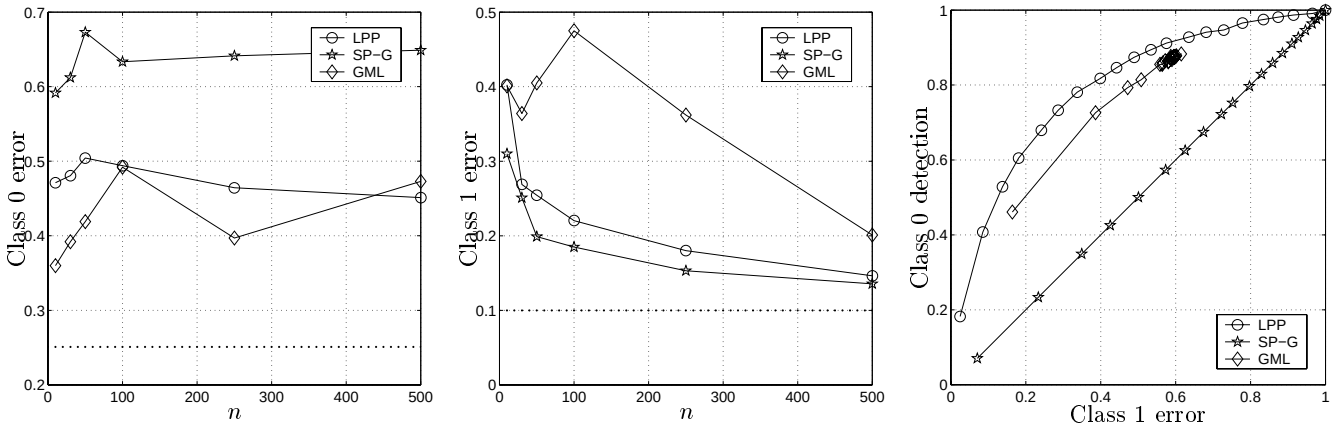
error, class 0 detection rate)= $(E_S[e_1(f)], 1 - E_S[e_0(f)])$ as an ROC. This plot shows that, with $n = 500$ samples and class 1 errors in the range approximately 0.3 to 0.9, SP outperforms GML which in turn is superior to LPP-G. Even though the GML performance curve is entirely above the LPP-G performance curve there are a wide range of false alarm rates achieved by LPP-G that cannot be achieved with GML. Figure 4 presents the results for the *I–I* distribution. While Figure 4(a) shows that GML outperforms the simple classes LPP and SP-G in terms of

class 0 error, Figure 4(b) shows just the opposite for class 1 error. The ROC plot in Figure 4(c) is qualitatively similar to that in Figure 3(c).



(a) Class 0 error comparison. (b) Class 1 error comparison (c) ROC Curve, $n = 500$

Figure 3: Experimental results for the Neyman–Pearson problem with the I–γI distribution. We compare GML, SP, and LPP-G. Figure 3(a) plots the class 0 errors and Figure 3(b) plots the class 1 errors verses training set size n. The Bayes' values are shown as a dark dotted line. Figure 3(c) plots the (false alarm rate, detection rate) = $(E_S[e_1(f)],\ 1 - E_S[e_0(f)])$ for training sets of size $n = 500$ as the value of α (the constraint on the class 1 empirical error) is varied from 0 to 1 in increments of 0.05.

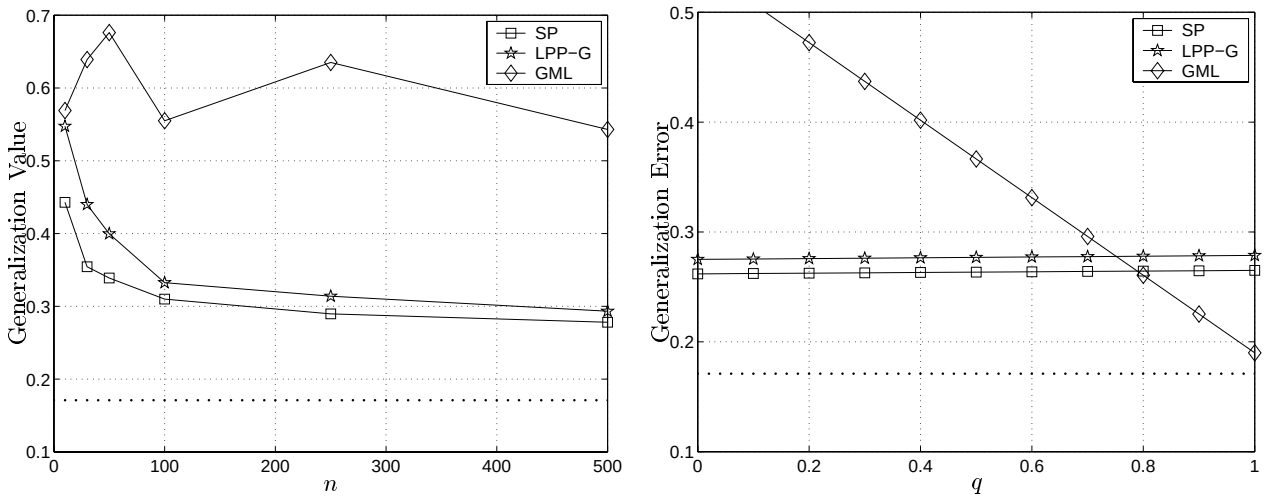


(a) Class 0 error comparison. (b) Class 1 error comparison (c) ROC Curve, $n = 500$

Figure 4: Experimental results for the Neyman–Pearson problem with the I–I distribution. We compare GML, LPP and SP-G. Figure 4(a) plots the class 0 errors and Figure 4(b) plots the class 1 errors verses training set size n. The Bayes' values are shown as a dark dotted line. Figure 4(c) plots the (false alarm rate, detection rate) = $(E_S[e_1(f)],\ 1 - E_S[e_0(f)])$ for training sets of size $n = 500$ as the value of α (the constraint on the class 1 empirical error) is varied from 0 to 1 in increments of 0.05.

We now describe the experimental results for the min–max problem. Figure 5 presents the results for the I–γI distribution. Figure 5(a) plots the generalization value as a function of

n for GML, SP and LPP-G. The performance of both simple classifiers is significantly better than GML. When a classifier designed with the min–max strategy is employed in practice its classification error depends on the value of the class marginal encountered in the real world environment. This value is unknown and in some cases may change with time. Thus, a common diagnostic for min–max is to plot the error $E_S[e(\hat{f}, q)]$ for the entire range of possible values of q from 0 to 1. Figure 5(b) plots the error $E_S[e(\hat{f}, q)]$ achieved by SP, LPP-G and GML with training size $n = 500$ as the class marginal $q = P(y = 0)$ ranges from 0 to 1. Since $E_S[e(\hat{f}, q)] = qE_S[e_0(\hat{f})] + (1-q)E_S[e_1(\hat{f})]$ these errors are linear functions of q. At the Bayes' optimal $e_0 = e_1 = 0.171$ and the error is the same for all values of q. For most values of q the error for the simple classifiers is superior to that for GML. Perhaps more importantly, the higher slope of the GML performance curve indicates that the GML solution lacks the robustness we seek with the min–max approach. Figure 6 presents the results for the I–I distribution where we compare GML, LPP and SP-G. These results show that GML is more competitive with the simple classifiers on this distribution. Indeed Figure 6(b) shows that GML is comparable to SP-G for small sample sizes and Figure 6(b) shows that the performance of GML falls between SP-G and LPP at $n = 500$.



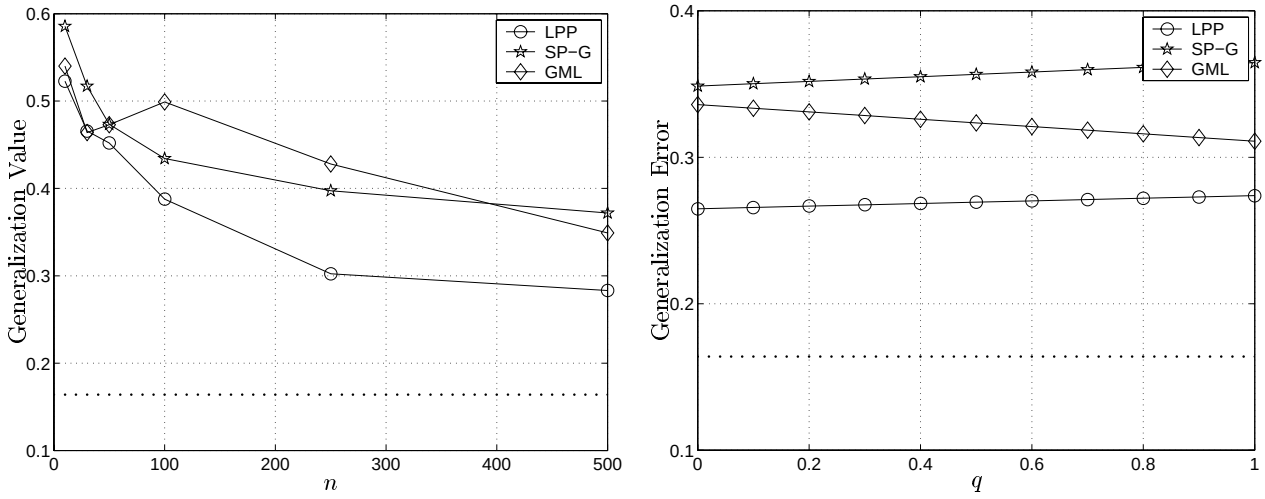
(a) Generalization Value vs. n

(b) Generalization Error, $n = 500$

Figure 5: Experimental results for the min–max problem with the I–γI distribution. We compare GML, SP and LPP-G. Figure 5(a) plots the generalization value verses training set size n. The Bayes' value is shown as a dark dotted line. Figure 5(b) plots the error $E_S[e(\hat{f}, q)]$ for q from 0 to 1 with training set size $n = 500$.

5.2 Real–World Data

This section describes experiments with three different real–world data sets. The first two are the *ionosphere* and *Pima Indian diabetes* data sets from the UCI repository (Blake & Merz, 1998) and the third is an *artificial nose* data set collected at Tufts University (Dickinson, White, Kauer, & Walt, 1996; White, Kauer, Dickinson, & Walt, 1996). The ionosphere data set consists of $n = 351$ samples of radar signals, $n_0 = 126$ with label $y = 0$ that passed through



(a) Generalization Value vs. n

(b) Generalization Error, $n = 500$

Figure 6: Experimental results for the min–max problem with the I–I distribution. We compare GML, LPP and SP-G. Figure 6(a) plots the generalization value verses training set size n. The Bayes' value is shown as a dark dotted line. Figure 6(b) plots the error $E_S[e(\hat{f}, q)]$ for q from 0 to 1 with training set size $n = 500$.

the ionosphere and $n_1 = 225$ with label $y = 1$ that did not pass through the ionosphere. Each radar signal contains $d = 34$ real valued measurements. The Pima Indian data set consists of $n = 768$ samples each containing $d = 8$ measurements. While it is often asserted that this data set has no missing values, Ripley (1996) comments that this is not true. Only the first of the 8 measurements can realistically take a value of 0, but many of the samples have value 0 for measurements 2 through 8. Ripley's recommendation is to exclude the 5th measurement, which is insulin level, because it has value 0 for so many of the samples. Then with the remaining 7 measurements he excludes samples which have value 0 for measurements 2 through 7. This reduces the data set to $n = 532$ samples with $d = 7$ measurements each. This reduced data set contains $n_0 = 355$ samples with label $y = 0$ (no diabetes) and $n_1 = 177$ samples with $y = 1$ (diabetes). The third data set is taken from an artificial nose developed at Tufts University. The nose consists of 19 optical fibers each of which has been coated on one end with a different organic dye. The data consists of the change in emission fluorescence intensity over time for each fiber. The change in intensity is measured at both the 620nm and 680nm wavelengths in each fiber. A 20 second time interval is sampled at 60 equally spaced points for each wavelength in each fiber. Hence each record consists of 38 observations each of which contains 60 samples, so each data sample contains $d = 2280$ measurements. Data samples were collected by exposing the fiber bundle to a 4 second pulse of a particular compound or mixture of compounds. The data set contains $n_0 = 760$ samples where the mixture contained trichloroethylene (TCE) and $n_1 = 352$ samples where the mixture contained no TCE, for a total of $n = 1112$ samples.

In the real–world environment associated with these data sets there is an obvious role for a two–class classifier, but the specific classifier design problem is not clear. In our first experiment we treat all three as error minimization problems and therefore assume that the data are gathered under the *i.i.d.* sample plan. This allows us to compare with generalization value

estimates reported elsewhere. In truth however we do not know how the data were gathered or whether error minimization is the appropriate design problem. In the artificial nose data set roughly 68% of the samples contain TCE. Although we do not know the probability that a sample will contain TCE, the value 0.68 seems unusually high and leads us to question the assumed *i.i.d.* sample plan. In contrast, the retrospective sample plan seems more plausible for this data and under this sample plan the min–max design problem with unknown $q = P(y = 0)$ seems more appropriate. Alternatively, it is not hard to imagine the TCE detector being used in an environment where it is important to control the false alarm rate below some fixed value, in which case the Neyman–Pearson design problem may be more appropriate. Thus, in our second experiment we use the nose data to obtain solutions to both the min–max and Neyman–Pearson problems under the assumption that the data are gathered retrospectively.

In these experiments we compare estimates of the average error values of the learning strategies. These estimates are obtained as follows. For each learning strategy the following is repeated $k = 100$ times for the ionosphere and diabetes data and $k = 25$ times for the nose data. The data set is randomly partitioned into two subsets: a training set containing approximately 2/3 of the data and a test set containing the remaining 1/3. When *i.i.d.* sampling is assumed the data set is partitioned as a whole using a random 2/3–1/3 split, and when retrospective sampling is assumed the data for each class is partitioned separately using a random 2/3–1/3 split. The learning strategy is applied to the training set and an estimate of the errors $e(\hat{f})$, $e_0(\hat{f})$ and $e_1(\hat{f})$ of the resulting classifier are computed using the test set. The estimated errors are averaged over the k runs to obtain estimates of the average generalization error for the error minimization and the Neyman–Pearson problem, and the average generalization error as a function of q for the min–max problem.

In our first experiment we compare learning strategies for the error minimization problem for all three data sets. For the ionosphere data Breiman (1999) reports a generalization value estimate of 5.5% for his random forest learning strategy. We implemented the same random forest learning strategy using Breiman's software and obtained an estimate of 6.6% with our error estimation procedure. On the reduced Pima Indian data set Ripley (1996) obtained a generalization value estimate of about 20% for the best classifier, and he performed a diagnostic which he claims lower bounds the Bayes value for this problem at about 15%. Using the nose data set Priebe (2001) reports an estimated generalization value of 12.5% for the best k-nearest-neighbor strategy, and 4.5% for a generalized Wilcoxon-Mann-Whitney classifier.

We report results for the five simple classifier learning strategies along with GML, SVM-SM and random forests (RF). For GML we used a regularization parameter $\gamma = 10^{-6}$. For support vector machines we used the SVM^{light} learning algorithm described in (Joachims, 1999) with default settings for the parameter values. All simple classifiers and support vector machines were employed in both their basic form and with the quadratic kernel $K(x_1, x_2) = \left(\langle x_1, x_2 \rangle + 1\right)^2$. All simple classifiers with quadratic kernels were boosted 500 rounds using the AdaBoost method in Freund and Shapire (1997). We used the random forest implementation Forest-RI from Breiman (1999) with number of trees $t = 1000$ and number of variables per node $F = 4$ for the ionosphere and diabetes data sets, and $t = 500$ and $F = 20$ for the nose data. The results are shown in Table 2.

As a general rule the simple linear classes performed better than the simple spherical classes (with no kernels or boosting). Indeed, the SC class does not perform well on any of the

	Ionosphere	Diabetes	Tuft's Nose
LPP	19.0/11.4/***6.50***	***23.7***/24.0/26.7	16.6/16.5/***9.8***
LPC	19.6/16.6/***8.60***	24.1/***23.2***/26.2	16.9/18.7/***13.6***
LCC	18.8/***14.9***/17.7	22.9/***22.6***/24.4	20.8/***20.0***/21.7
SP	26.8/28.3/***8.09***	24.5/***24.1***/26.3	18.2/20.6/***15.1***
SC	26.6/***24.8***/29.4	33.1/***32.5***/33.9	***32.5***/32.5/33.0
SVM-SM	13.0/***9.1***	22.8/***22.7***	***16.8***/16.9
GML	12.5	23.9	-
RF	6.60	23.2	10.1

Table 2: Estimates of the average generalization value for LPP, LPC, LCC, SP, SC, Gaussian maximum likelihood (GML), support vector machines (SVM-SM) and random forests (RF). The simple classes have three entries A/B/C where A is the value for the basic class, B is the value for the class with quadratic kernel $K(x_1, x_2) = \left(\langle x_1, x_2\rangle + 1\right)^2$ and C is the value for the class with quadratic kernel and 500 rounds of boosting (except for the LPP class with the Tuft's nose data where 250 rounds of boosting are used). The reported results are for the final round of boosting. The best of the three entries A/B/C is shown in bold italic font. The SVM-SM has two entries A/B where A is the linear support vector machine and B is the support vector machine with quadratic kernel. Again, the best of the two entries A/B is shown in bold italic font. No result is provided for GML on the Tuft's nose data because the dimension of the data is too high to obtain meaningful results with this method.

three data sets. In most cases the incorporation of kernels and/or boosting lead to improved performance for the simple classes, except for the diabetes data where almost all the methods perform similarly. Many of the boosted simple classifiers perform very well compared to SVM-SM and GML. In addition there is always at least one simple class (with kernels or boosting) whose performance is comparable to the best overall. The best performances across the two most challenging data sets, Ionosphere and nose, are provided by LPP with boosting and RF.

It is instructive to consider what can be achieved with the nose data by the the restricted class LPP compared to traditional linear classifiers. It is no surprise that this data set of 1112 samples in 2280 dimensions is linearly separable. Nearly all training algorithms for traditional linear classifiers are designed to produce a separating solution when one exists. Of all the linear classifiers that separate the data the SVM-SM solution is generally considered among the best. On the other hand the LPP solution, which does not separate the data, provides almost identical performance. The LPP strategy has the additional benefit that it has a lower error deviance which means that estimates of its performance are more accurate. The nonseparability of the LPP solution also means that boosting can be employed to improve performance, which is not true for traditional linear classifier learning strategies that produce separating solutions [8]. This is significant in the nose data because boosting LPP leads to a performance that is significantly better than SVM-SM.

[8] Boosting effectively terminates on the first round when the base classifier achieves zero training error.

In our next experiment we obtain solutions to the Neyman–Pearson and min–max problems using the nose data. The dimension $d = 2280$ of this data is too high to obtain meaningful results with the GML method. Further, since we know of no boosting strategy for the Neyman–Pearson or min–max problems, and the results in Table 2 suggest that LPP without a kernel will work well, we employ only the basic LPP class.

We start with the Neyman–Pearson problem. To obtain a receiver operating curve (ROC) we employed the Neyman–Pearson learning strategy varying α from 0 to 1 in increments of 0.05, with $\epsilon_1 = 0.002$ in each case. Figure 7 plots the corresponding sequence of two-tuples (class 1 error, class 0 detection rate)= $(E_S[e_1(f)], 1 - E_S[e_0(f)])$ as an ROC. The detection rate starts to decrease rapidly when the false alarm rate falls below 0.4, and even more rapidly when it falls below 0.2. The $*$ symbol in Figure 7 represents the average operating point for classifiers designed to solve the error minimization problem treated earlier. The detection rate for this operating point is above 90%, but the false alarm rate is nearly 40%. Although we do not know the application well it seems plausible that a false alarm rate of 40% is unacceptable. In contrast, employing the Neyman–Pearson learning strategy allows one to operate at a point on the ROC curve which has a reduced false alarm rate.

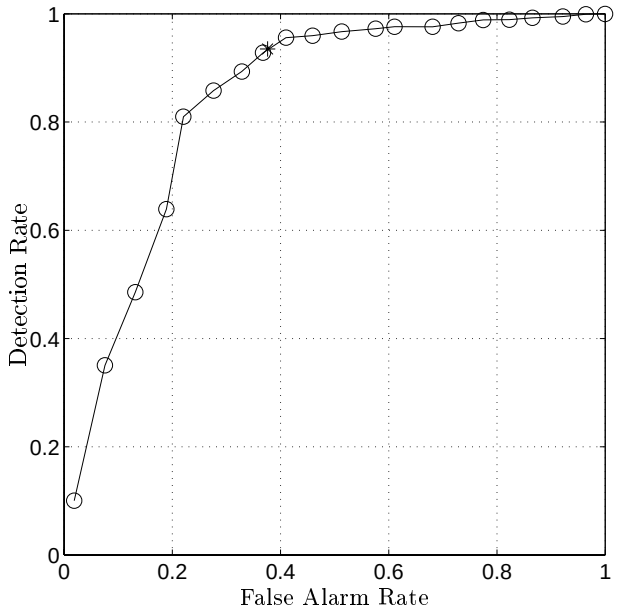


Figure 7: The ROC curve for the Neyman–Pearson learning strategy applied to the nose data with the LPP class. The plot shows the false alarm rate $E_S[e_1(f)]$ verses the detection rate $1 - E_S[e_0(f)]$ as α is varied from 0 to 1 in increments of 0.05 (with $\epsilon_1/2 = 0.001$ in each case). The $*$ symbol represents the average operating point for classifiers designed to solve the error minimization problem.

The results for the min–max problem are illustrated in Figure 8 which shows three curves. The curve labeled MM is the average generalization error $E_S[e(\hat{f}, q)]$ as a function of q for the min–max learning strategy. To investigate the price we pay for not knowing q we perform the following experiment. We assume $q = P(y = 0)$ is known and employ an empirical error minimization learning strategy with LPP on the retrospectively sampled data weighted to

simulate the marginal q. In particular we set the weights to $w_n(i) = qn/n_0$ for samples with $y_n(i) = 0$ and to $w_n(i) = (1-q)n/n_1$ for samples with $y_n(i) = 1$. The curve labeled EMQ is (an estimate of) the average generalization error achieved by this strategy [9]. The gap between the MM and EMQ curves represents the difference in performance due to lack of knowledge of q. Since the min–max classifier is designed to perform well for the worst case value of q it is no surprise that the gap between the MM and EMQ curves is larger near the best case values of q which occur at the extremes.

Now suppose that an incorrect value of the marginal is assumed and we employ the error minimization strategy above. In particular suppose the restrospectively sampled data is incorrectly assumed to be *i.i.d.* from the joint distribution and we employ the error minimization strategy. If q actually turns out to be $n_0/n = 0.68$ (as it is in the nose data) then the average generalization error for this approach is approximately 16.6%. We plot this value at $q = 0.68$ using the symbol $*$ in Figure 8. The performance of the min–max learning strategy (the MM curve) is nearly constant over q and its error is approximately 5% higher than this value 16.6%. On the other hand the average generalization error $E_S[e(\hat{f}, q)]$ for classifiers designed using the error minimization learning strategy is illustrated by the curve labeled EM. When q is less than approximately 0.53 the average performance of classifiers designed using the error minimization learning strategy is worse than the average performance of classifiers designed using the min–max learning strategy. The reverse is true when q is greater than 0.53. In this problem we do not know the probability that a sample will contain TCE, but it seems likely that it is much less than 0.53 in which case the performance of the min–max solution will be much better than the error minimization solution.

References

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases.* http://www.ics.uci.edu/∼mlearn/MLRepository.html: University of California, Irvine, Dept. of Information and Computer Sciences.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, *2*, 499–526.

Breiman, L. (1999). *Random forests* (Technical Report No. 567). Berkeley, CA: University of California - Berkeley.

Campbell, C., & Cristianini, N. (1999). *Simple training algorithms for support vector machines* (Technical Report CIG-TR-KA). University of Bristol, Engineering Mathematics, Computational Intelligence Group.

Cannon, A., Ettinger, M., Hush, D., & Scovel, C. (2002a). Machine learning with data dependent hypothesis classes. *Journal of Machine Learning Research*, *2*, 335–358.

Cannon, A., Howse, J., Hush, D., & Scovel, C. (2002b). *Learning with the Neyman-Pearson and min-max criteria* (Los Alamos Technical Report Nos. LA–

[9] To obtain these estimates we use the same cross validation procedure described earlier for estimating the average generalization error when the data is gathered under the retrospective sample plan.

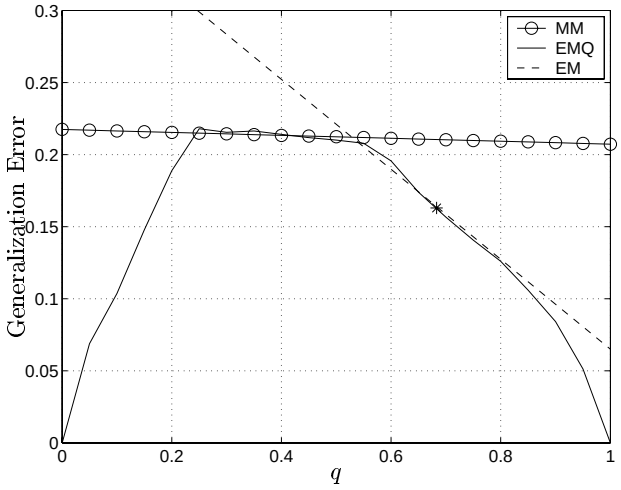


Figure 8: Results for the min–max experiments applied to the nose data with the LPP class. The MM curve shows the average generalization error $E_S[e(\hat{f}, q)]$ verses $q = P(y = 0)$ for the min–max learning strategy. The EMQ curve shows the average generalization error achieved by an empirical error minimization learning strategy that knows the value of q. The $*$ symbol is the average generalization error for classifiers designed using the error minimization learning strategy when $q = 0.68$. The EM curve is the average generalization error $E_S[e(\hat{f}, q)]$ as a function of q for classifiers designed using the error minimization learning strategy when $n_0/n = 0.68$.

UR–02–2951). Los Alamos National Laboratory. (submitted for publication, http://wwwc3.lanl.gov/ml/pubs_select.shtml)

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods* (1st ed.). Cambridge ; United Kingdom: Cambridge University Press.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition.* New York, NY: Springer.

Dickinson, T., White, J., Kauer, J., & Walt, D. (1996). A chemical–detecting system based on a cross–reactive optical sensor array. *Nature*, *382*, 697–700.

Freund, Y., & Shapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition* (2nd ed.). San Diego, CA: Academic Press.

Huber, P. (1981). *Robust statistics.* New York: John Wiley & Sons, Inc.

Hush, D., & Scovel, C. (2003). Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, *51*, 51–71.

Joachims, T. (1999). Making large–scale SVM learning practical. In B. Schlkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - support vector learning.* MIT Press.

Priebe, C. (2001). Olfactory classification via interpoint distance analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(4), 404–413.

Ripley, B. (1996). *Pattern recognition and neural networks.* Cambridge, UK: Cambridge University Press.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*(5), 1651–1686.

Van-Trees, H. (1968). *Detection, estimation and modulation theory: Part 1.* New York, NY: John Wiley & Sons, Inc.

Vapnik, V. N. (1998). *Statistical learning theory.* New York NY: John Wiley & Sons, Inc.

Vapnik, V. N., & Chervonenkis, A. (1974). *Theory of pattern recognition.* Moscow: Nauka. ((in Russian))

White, J., Kauer, J., Dickinson, T., & Walt, D. (1996). Rapid analyte recognition in a device based on optical sensors and the olfactory system. *Analytical Chemistry*, *68*, 2191–2202.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems 13* (pp. 682–688). MIT Press.